IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-6, Issue-2 E-ISSN: 2347-2693

Optimization of Diabetes Training DATA using Machine Learning Algorithms

M. Samba Siva Rao[1*], M.Yaswanth[2], K. Raghavendra swamy [3]

[1*]Dept. Of CSE, Usha Rama college of engineering and Technology, Tealaprolu, JNTUK, INDIA
[2] Dept. of CSE, PVP Siddhartha college of engineering, Vijayawada, JNTUK, INDIA
[3] Dept. Of IT, BDPS IT Training and Data Processing Solutions, Vijayawada, INDIA

[*]*Corresponding Author: csehod@usharama.in*

Available online at: www.ijcseonline.org

Received: 24/Jan/2017, Revised: 6/Feb/2017, Accepted: 18/Feb/2017, Published: 28/Feb/2017

Abstract- Diabetes Disease is one of most common disease in our modern life, and in this paper we are using different Super vised and un Super vised Machine learning Algorithms to Analyze and optimize accuracy of Training Data and classify , diagnosis , accuracy of Algorithms with python Machine learning modules like pandas, sklearn, Seaborn.

Keywords: python, Machine Learning, Pandas, Seaborn, Sklearn, Diabetes

I. INTRODUCTION

Diabetes has been recognized as a continuing health challenge for the twenty-first century, both in developed and developing countries. It is understood that diabetes prevalence is increased because of modern lifestyles, urbanization, and economic development. It is a global problem with devastating human, social, and economic impact, affecting around 300 million people worldwide.

To diagnose diabetes, a physician has to analyze many factors. Undoubtedly, the evaluations of data obtained from patients and expert decisions are critical for diagnosis. However, factors such as lack of experience by the experts, or their fatigue, may lead to erroneous diagnosis. Early intervention with lifestyle modifications or pharmacotherapy has been shown to effectively delay or prevent type 2 diabetes and its complications in adults .

II. RELATED WORK

We have different Naïve Bayes and Decision Tree Algorithms for diagnosis Diabetic patient data, in this paper we evaluate different Machine learning Algorithms to optimize the accuracy of the end result

III. METHODOLOGY

Machine learning is the scientific field dealing with the ways in which machines learn from experience. For many scientists, the term “machine learning” is identical to the term “artificial intelligence”, given that the possibility of learning is the main characteristic of an entity called intelligent in the broadest sense of the word. The purpose of machine learning is the construction of computer systems that can adapt and learn from their experience

Machine learning tasks are typically classified into three broad categories These are: a) supervised learning, in which the system infers a function from labeled training data, b) unsupervised learning, in which the learning system tries to infer the structure of unlabeled data, and c) reinforcement learning.

IV. RESULTS AND DISCUSSION

An Overview of Python:

we are using in this problem model python tools and modules , python is open source programming tool software , it supports lot of features and modules to handle data analytics, machine learning and statics problems easily with simpler functional support , writing and debugging program in python were very easy and implementation of scientific problems. here in this problem model we are using matplotlib, numpy and pyplots for plotting output results, also supports Machine learning Algorithms like Classification, KNN classifications, Linear Regression and other Algorithms.

An overview of Problem:

In this paper we are using different Machine learning Algorithms and analyzing accuracy reports

Classify the training data depending on outcome of diabetes 1 / 0
Outcome" is the feature we are going to predict, 0 means No diabetes, 1 means diabetes. Of these 795 data points, 509 are labeled as 0 and 286 as 1

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
diabetes = pd.read_csv('diabetes.csv')
import seaborn as sns
sns.countplot(diabetes['Outcome'],label="Count")
```

i.e. 509 classified as non-diabetic and 286 classified as diabetic

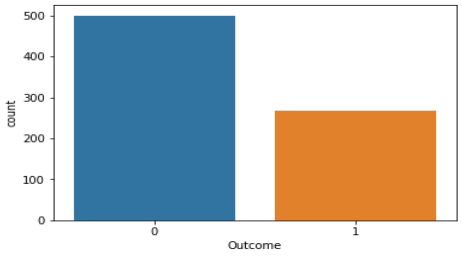

Fig.1

k-Nearest Neighbors:

The k-NN algorithm [4]is arguably the simplest machine learning algorithm. Building the model consists only of storing the training data set. To make a prediction for a new data point, the algorithm finds the closest data points in the training data set—its "nearest neighbors."[1][2]

First, Let's investigate whether we can confirm the connection between model complexity and accuracy:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
train_test_split(diabetes.loc[:, diabetes.columns !=
'Outcome'], diabetes['Outcome'],
stratify=diabetes['Outcome'], random_state=66)
from sklearn.neighbors import KNeighborsClassifier
training_accuracy = []
test_accuracy = []
# try n_neighbors from 1 to 10
neighbors_settings = range(1, 11)
for n_neighbors in neighbors_settings:
    # build the model
    knn = KNeighborsClassifier(n_neighbors=n_neighbors)
    knn.fit(X_train, y_train)
    # record training set accuracy
    training_accuracy.append(knn.score(X_train, y_train))
    # record test set accuracy
    test_accuracy.append(knn.score(X_test, y_test))
plt.plot(neighbors_settings, training_accuracy,
label="training accuracy")
plt.plot(neighbors_settings, test_accuracy, label="test
accuracy")
plt.ylabel("Accuracy")
plt.xlabel("n_neighbors")
plt.legend()
plt.savefig('knn_compare_model')
```

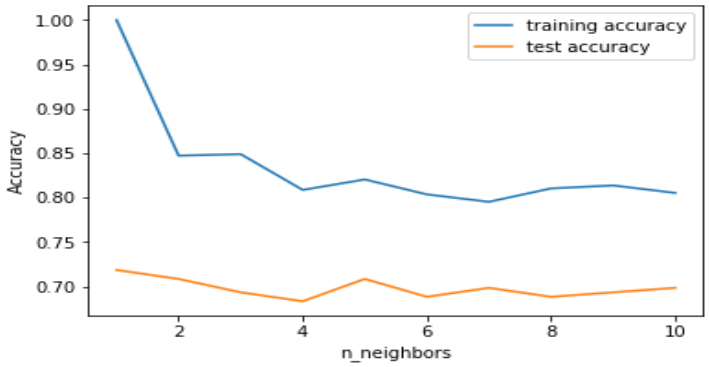

Fig.2

The above plot shows the training and test set accuracy on the y-axis against the setting of neighbor's on the x-axis. Considering if we choose one single nearest neighbor, the prediction on the training set is perfect. But when more neighbors are considered, the training accuracy drops, indicating that using the single nearest neighbor leads to a model that is too complex. The best performance is somewhere around 9 neighbors.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
diabetes = pd.read_csv('diabetes.csv')
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test =
train_test_split(diabetes.loc[:, diabetes.columns !=
'Outcome'], diabetes['Outcome'],
stratify=diabetes['Outcome'], random_state=66)
from sklearn.neighbors import KNeighborsClassifier
training_accuracy = []
test_accuracy = []
knn = KNeighborsClassifier(n_neighbors=9)
knn.fit(X_train, y_train)
print('Accuracy of K-NN classifier on training set:
{:.2f}'.format(knn.score(X_train, y_train)))
print('Accuracy of K-NN classifier on test set:
{:.2f}'.format(knn.score(X_test, y_test)))
```


We classify by using neighbor classifier on value 9 we got classification according to outcome and not outcome as below

Accuracy of K-NN classifier on training set: 0.81
Accuracy of K-NN classifier on test set: 0.69

Regression:

Logistic Regression [4][6][1]is one of the most common classification algorithms using logistic regression for this problem is that the values of the dependent variable, while represented by "1" and "0".

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
diabetes = pd.read_csv('diabetes.csv')
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(diabetes.loc[:, diabetes.columns != 'Outcome'], diabetes['Outcome'], stratify=diabetes['Outcome'], random_state=66)
from sklearn.neighbors import KNeighborsClassifier
training_accuracy = []
test_accuracy = []
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression().fit(X_train, y_train)
print "Training set score: {:.3f}".format(logreg.score(X_train, y_train))
print "Test set score: {:.3f}".format(logreg.score(X_test, y_test))
```

Training set score: 0.767
Test set score: 0.779

Decision Tree:

Decision tree[4][5] learning is a method commonly used in data mining. The goal is to create a model that predicts the value of a target variable based on several input variables.. Each interior node corresponds to one of the input variables; there are edges to children for each of the possible values of that input variable. Each leaf represents a value of the target variable given the values of the input variables represented by the path from the root to the leaf.

```
from sklearn.tree import DecisionTreeClassifier
tree = DecisionTreeClassifier(random_state=0)
tree.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(tree.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(tree.score(X_test, y_test)))
```

Accuracy on training set: 1.000
Accuracy on test set: 0.729

The accuracy on the training set is 100%, while the test set accuracy is much worse. This is an indicative that the tree is overfitting and not generalizing well to new data. Therefore, we need to apply pre-pruning to the tree.

We set max_depth=3, limiting the depth of the tree decreases overfitting. This leads to a lower accuracy on the training set, but an improvement on the test set.

Accuracy on training set: 0.772
Accuracy on test set: 0.693

Feature Importance in Decision Trees
Feature importance rates how important each feature is for the decision a tree makes. It is a number between 0 and 1 for each feature, where 0 means “not used at all” and 1 means “perfectly predicts the target”. The feature importances always sum to 1:

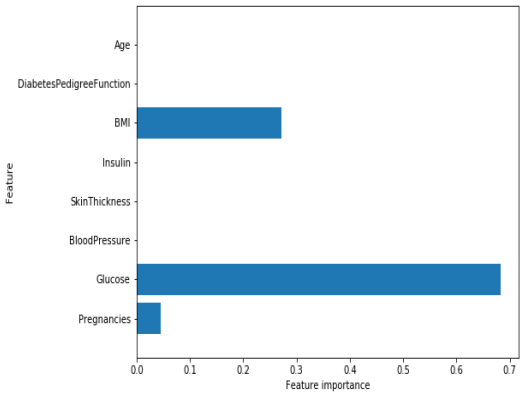

Fig.3

Random Forest

Random forest algorithm is a supervised classification algorithm. As the name suggest, this algorithm creates the forest with a number of trees.

In general, the more trees in the forest the more robust the forest looks like. In the same way in the random forest classifier, the higher the number of trees in the forest gives the high accuracy results.

Let's apply a random forest consisting of 100 trees on the diabetes data set:

```
from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(n_estimators=100, random_state=0)
rf.fit(X_train, y_train)
print("Accuracy on training set: {:.3f}".format(rf.score(X_train, y_train)))
print("Accuracy on test set: {:.3f}".format(rf.score(X_test, y_test)))
```

Accuracy on training set: 1.000
Accuracy on test set: 0.759

The random forest gives us an accuracy of 78.6%, better than the logistic regression model or a single decision tree, without tuning any parameters. However, we can adjust the max_ features setting, to see whether the result can be improved.

Accuracy on training set: 0.810
Accuracy on test set: 0.734

it did not, this indicates that the default parameters of the random forest work well.

Support Vector Machine:

In machine learning, support vector machines (SVMs, also support vector networks) are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier .

```
from sklearn.svm import SVC
svc = SVC()
svc.fit(X_train, y_train)
print("Accuracy on training set: {:.2f}".format(svc.score(X_train, y_train)))
print("Accuracy on test set: {:.2f}".format(svc.score(X_test, y_test)))
```

Accuracy on training set: 1.00
Accuracy on test set: 0.69

Analysis of optimization of different Machine learning Algorithm:

Machine Algorithm	Accaracy
k-Nearest Neighbors	Accuracy of K-NN classifier on training set: 0.81
	Accuracy of K-NN classifier on test set: 0.69
KNN classifier	Accuracy of K-NN classifier on training set: 0.81 Accuracy of K-NN classifier on test set: 0.69
Logistic Regression	Training set score: 0.767 Test set score: 0.779
Decision Tree	Accuracy on training set: 1.000 Accuracy on test set: 0.729
Random Forest	Accuracy on training set: 1.000 Accuracy on test set: 0.759
Support Vector Machine	Accuracy on training set: 1.00 Accuracy on test set: 0.69

V. CONCLUSION AND FUTURE SCOPE

In this paper we analyze the diabetes data and compare the optimizations using different Machine learning algorithms. We can also implement different output accuracy in sikit learn modules.

REFERENCES

[1]. Sikit learn cook Book
[2]. https://www.kdnuggets.com/2015/11/seven-steps-machine-learning- python.html
[3]. Understanding Machine Learning: From Theory to Algorithms By Shai Shalev-Shwartz and Shai Ben-David
[4]. Machine Learning Yearning By Andrew Ng
[5]. Introduction to Machine Learning with Python: A Guide for Data Scientists Book by Andrea C. Müller
[6]. Machine Learning in Python: Essential Techniques for Predictive Analysis
[7]. https://towardsdatascience.com/machine-learning-for-diabets

