AI BASED DIABETES PREDICTION SYSTEM

USING MACHINE LEARNING ALGORITHM

Phase 4: In this phase, we'll continue building the diabetes prediction system by:

- Selecting a machine learning algorithm
- Training the model
- Evaluating its performance

Dataset: https://www.kaggle.com/datasets/mathchi/diabetes-data-set

ALGORITHM SELECTION

The algorithm selected are :

1. Random Forest
2. Decision Tree
3. Support Vector Machine (SVM)

RANDOM FOREST:

Random Forest is a machine learning ensemble algorithm that combines multiple decision trees to make more accurate predictions. It works by constructing a forest of decision trees during training and averaging their predictions for better overall results. It's known for its robustness, versatility, and ability to handle both classification and regression tasks.

DECISION TREE:

Decision tree is a popular machine learning algorithm used for classification and regression tasks. It models decisions and their possible consequences in a tree-like structure. At each node of the tree, a decision is made based on a feature, leading to one of several possible outcomes. This process continues until a final decision or prediction is reached. Decision trees are interpretable, easy to understand, and can handle both categorical and numerical data. Popular variations include Rand

SUPPORT VECTOR MACHINE (SVM):

A Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification and regression. It works by finding a hyperplane that best separates different classes in the input data while maximizing the margin between them. The data points closest to this hyperplane are called support vectors. SVMs can handle high-dimensional data and are effective for both linear and non-linear problems through the use of kernel functions. They are particularly useful when dealing with binary classification tasks.

MODEL BUILDING :

Spitting the Dataset:

```
#Splitting the dataset

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

from sklearn.model_selection import train_test_split, GridSearchCV,
cross_val_score

X_train, X_test, y_train, y_test = train_test_split(X,y,
test_size=0.33,
                                                    random_state=7)

#Check columns with zero values - checking this time so that right
data should go for model training

print("Total number of rows: {0}", format(len(diabetes_df)))
print("Number of rows missing Pregnancies: {0}",
      format(len(diabetes_df.loc[diabetes_df['Pregnancies']==0])))
print("Number of rows missing Glucose: {0}"
      , format(len(diabetes_df.loc[diabetes_df['Glucose']==0])))
print("Number of rows missing BloodPressure: {0}",
      format(len(diabetes_df.loc[diabetes_df['BloodPressure']==0])))
print("Number of rows missing SkinThickness: {0}",
      format(len(diabetes_df.loc[diabetes_df['SkinThickness']==0])))
print("Number of rows missing Insulin: {0}",
      format(len(diabetes_df.loc[diabetes_df['Insulin']==0])))
print("Number of rows missing BMI: {0}",
      format(len(diabetes_df.loc[diabetes_df['BMI']==0])))
print("Number of rows missing DiabetesPedigreeFunction: {0}",

format(len(diabetes_df.loc[diabetes_df['DiabetesPedigreeFunction']==0]
```

RANDOM FOREST

```
)))
print("Number of rows missing Age: {0}",
format(len(diabetes_df.loc[diabetes_df['Age']==0])))
```

```
Total number of rows: {0} 768
Number of rows missing Pregnancies: {0} 111
Number of rows missing Glucose: {0} 5
Number of rows missing BloodPressure: {0} 35
Number of rows missing SkinThickness: {0} 227
Number of rows missing Insulin: {0} 374
Number of rows missing BMI: {0} 11
Number of rows missing DiabetesPedigreeFunction: {0} 0
Number of rows missing Age: {0} 0
```

```
#Imputing zeros values in the dataset

from sklearn.impute import SimpleImputer
import numpy as np

fill_values = SimpleImputer(missing_values=0, strategy='mean')
X_train = fill_values.fit_transform(X_train)
X_test = fill_values.fit_transform(X_test)

#Builidng the model using RandomForest

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n_estimators=200)
rfc.fit(X_train, y_train)
```

```
RandomForestClassifier(n_estimators=200)
```

```
# On training data
rfc_train = rfc.predict(X_train)
from sklearn import metrics

print("Accuracy_Score =", format(metrics.accuracy_score(y_train,
rfc_train)))
```

```
Accuracy_Score = 1.0
```

```
predictions = rfc.predict(X_test)

#Getting the accuracy score for Random Forest

from sklearn import metrics

print("Accuracy_Score =", format(metrics.accuracy_score(y_test,
predictions)))
```

```
Accuracy Score = 0.7440944881889764
```

DECISION TREE:

```
weighted avg       0.74      0.74      0.74       254
```

```
#Building the model using Support Vector Machine (SVM)

from sklearn.svm import SVC

svc_model = SVC()
svc_model.fit(X_train, y_train)
```

```
SVC()
```

```
#Predict
svc_pred = svc_model.predict(X_test)

#Accuracy score for SVM
from sklearn import metrics

print("Accuracy Score =", format(metrics.accuracy_score(y_test,
svc_pred)))
```

```
Accuracy Score = 0.7401574803149606
```

```
#Metrics for SVM
from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test, svc_pred))
print(classification_report(y_test,svc_pred))
```

```
[[143  19]
 [ 47  45]]
              precision    recall  f1-score   support

           0       0.75      0.88      0.81       162
           1       0.70      0.49      0.58        92

    accuracy                           0.74       254
   macro avg       0.73      0.69      0.69       254
weighted avg       0.73      0.74      0.73       254
```

SUPPORT VECTOR MACHINE (SVM):

```
from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))
```

```
[[132  30]
 [ 35  57]]
              precision    recall  f1-score   support

           0       0.79      0.81      0.80       162
           1       0.66      0.62      0.64        92

    accuracy                           0.74       254
   macro avg       0.72      0.72      0.72       254
weighted avg       0.74      0.74      0.74       254
```

```
#Building the model using DecisionTree

from sklearn.tree import DecisionTreeClassifier

dtree = DecisionTreeClassifier()
dtree.fit(X_train, y_train)
```

```
DecisionTreeClassifier()
```

```
predictions = dtree.predict(X_test)

#Getting the accuracy score for Decision Tree

from sklearn import metrics

print("Accuracy Score =",
format(metrics.accuracy_score(y_test,predictions)))
```

```
Accuracy Score = 0.7283464566929134
```

```
from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test, predictions))
print(classification_report(y_test,predictions))
```

```
[[124  38]
 [ 31  61]]
              precision    recall  f1-score   support

           0       0.80      0.77      0.78       162
           1       0.62      0.66      0.64        92

    accuracy                           0.73       254
   macro avg       0.71      0.71      0.71       254
```

CONCLUSION:

Therefore, Random forest is the best model for this prediction since it has an accuracy score of 0.76.