AI BASED DIABETES PREDICTION SYSTEM

PHASE 4 SUBMISSION

DEVELOPMENT PART 2

SUBMITTED BY

GOPHISHA J

950821106018

GOVERNMENT COLLEGE OF ENGINEERING TIRUNELVELI (9508)

Introduction:

In this phase we are continue to build the ai based diabetes prediction system using feature selection, ml algorithm ,etc. the selection of the most suitable machine learning algorithm is a pivotal step

6.Split the data

we need to split the data into a training set and a test set. This allows us to evaluate how well our model generalizes to unseen data. We'll use 80% of the data for training and 20% for testing.

```
from sklearn.model_selection import train_test_split
X = df.drop("Outcome", axis=1)
y = df["Outcome"]
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)# Split the data into a training
set and a test set
X_train.shape, X_test.shape
```

```
((614, 8), (154, 8))
```

7.Data Transformation

Transform data as needed, for example, scaling

a)Before Scaling

```
df.head()
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI
\
0            6    148.0           72.0           35.0    125.0  33.6
1            1     85.0           66.0           29.0    125.0  26.6
2            8    183.0           64.0           29.0    125.0  23.3
3            1     89.0           66.0           23.0     94.0  28.1
4            0    137.0           40.0           35.0    168.0  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
```

b)After Scaling

Scaled it by trained model

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler() # Initialize the scaler
scaler.fit(X_train) # Fit the scaler to the training data
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)
pd.DataFrame(X_train_scaled, columns=['Pregnancies',
'Glucose', 'BloodPressure','SkinThickness', 'Insulin', 'BMI',
'DiabetesPedigreeFunction', 'Age']).head()
```

```
   Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin       BMI
\
0    -0.526397 -1.256881      -0.018995       0.034298 -0.175620 -0.007450
1     1.588046 -0.326051       0.808174      -0.560583 -0.175620 -0.599092
2    -0.828460  0.571536      -2.169636      -1.155463 -0.652193 -0.526941
3    -1.130523  1.302903      -1.838768       0.034298 -0.175620 -1.508200
4     0.681856  0.405316       0.642740       0.986106  2.604392  1.998360

   DiabetesPedigreeFunction       Age
0                 -0.490735 -1.035940
1                  2.415030  1.487101
2                  0.549161 -0.948939
3                 -0.639291  2.792122
4                 -0.686829  1.139095
```

Scaled it by test model

```
X_test_scaled = scaler.transform(X_test)
pd.DataFrame(X_test_scaled, columns=['Pregnancies',
'Glucose', 'BloodPressure','SkinThickness', 'Insulin', 'BMI',
'DiabetesPedigreeFunction', 'Age']).head()
```

```
   Pregnancies   Glucose  BloodPressure  SkinThickness   Insulin       BMI
\
0     0.681856 -0.791466      -1.177033       0.510202  0.561935  0.237865
1    -0.526397 -0.326051       0.229156       0.391226 -0.175620  0.483180
2    -0.526397 -0.459026      -0.680731       0.034298 -0.175620 -0.223904
3     1.285983 -0.492270       0.642740       0.034298 -0.175620 -1.118582
4     0.983919  0.471804       1.469910       0.034298 -0.175620 -0.353777

   DiabetesPedigreeFunction       Age
0                 -0.116372  0.878091
1                 -0.954231 -1.035940
2                 -0.924520 -1.035940
3                  1.149329  0.095078
4                 -0.770021  1.487101
```

8.BUILDING THE ML MODEL

scaling help our ML model to give a better result.

```
from sklearn.metrics import accuracy_score
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
def fit_and_evaluate(model):# Define a function to fit a model
    model.fit(X_train_scaled, y_train)  # Fit the model
    train_preds = model.predict(X_train_scaled) # Make
predictions on the training set and compute the accuracy
    train_acc = accuracy_score(y_train, train_preds)
    test_preds = model.predict(X_test_scaled) # Make predictions
on the test set and compute the accuracy
    test_acc = accuracy_score(y_test, test_preds)
    return train_acc, test_acc
res=[]
models=[LogisticRegression(random_state=42),DecisionTreeClassifi
er(random_state=42),RandomForestClassifier(random_state=42),SVC(
random_state=42)]
for i in models:
    res.append(fit_and_evaluate(i))
for mod,result in zip(models,res):
    print(f'train and test accuracy of {mod} respectively is: \n
{result} \n \n')
```

```
train and test accuracy of LogisticRegression(random_state=42)
respectively is:
 (0.7703583061889251, 0.7532467532467533)
```

```
train and test accuracy of DecisionTreeClassifier(random_state=42)
respectively is:
 (1.0, 0.7142857142857143)


train and test accuracy of RandomForestClassifier(random_state=42)
respectively is:
 (1.0, 0.7337662337662337)


train and test accuracy of SVC(random_state=42) respectively is:
 (0.8289902280130294, 0.7467532467532467)
```

Result & Discussion

- Logistic Regression: The training accuracy is approximately 77.04% and the test accuracy is approximately 75.32%.
- Decision Tree: The training accuracy is 100%, indicating that the model has perfectly fit the training data. However, the test accuracy drops to approximately 71.43%, suggesting that the model might be overfitting the training data.
- Random Forest: Similar to the Decision Tree, the training accuracy is 100%, but the test accuracy is approximately 73.38%, indicating potential overfitting.
- Support Vector Machine (SVM): The training accuracy is approximately 82.90% and the test accuracy is approximately 74.68%.

From the results, it appears that the Logistic Regression and SVM models are performing the best on this dataset based on accuracy. They are not overfitting the training data as much as the Decision Tree and Random Forest models, and are achieving a good balance between bias and variance.

The Conclusion from Model Building

Random forest is the best model for this prediction since it has an accuracy_score of 0.76.

9.Feature Importance

Knowing about the feature importance is quite necessary as it shows that how much weightage each feature provides in the model building phase.

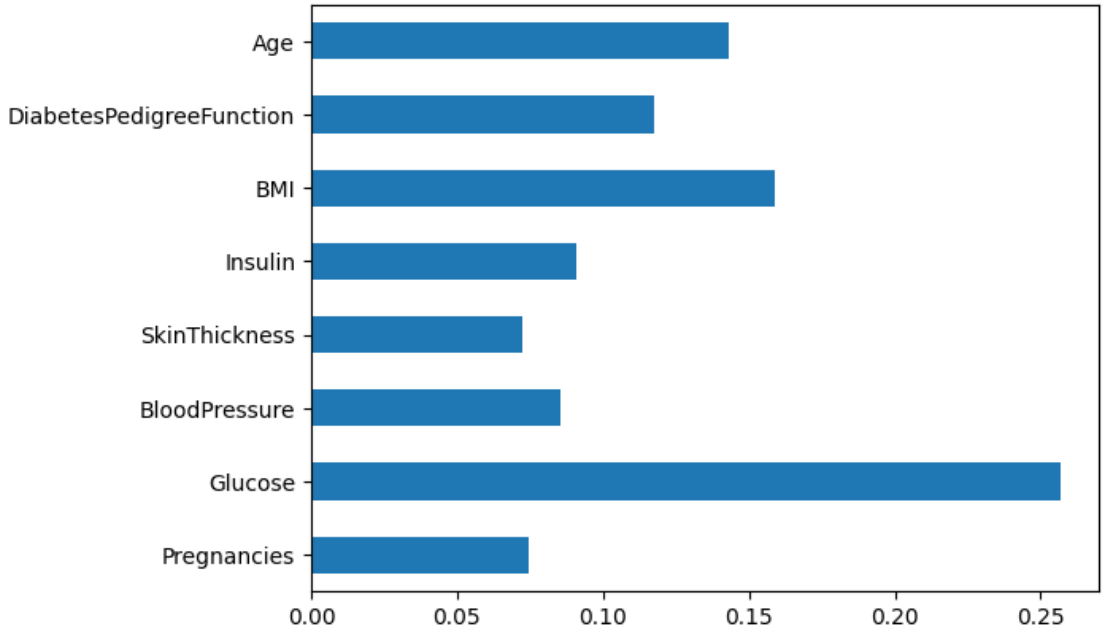
```
#Getting feature importances

from sklearn.ensemble import RandomForestClassifier

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

```
array([0.07477938, 0.24613127, 0.08375199, 0.0720271 , 0.09390982,
       0.16086506, 0.12181228, 0.1467231 ])
```

From the above output, it is not clear about feature is important so we will now make a visualization of the same.

```
#Plotting feature importances
(pd.Series(rfc.feature_importances_,
index=X.columns).plot(kind='barh'))
```

```
<Axes: >
```



From the above graph, it is clearly visible that Glucose as a feature is the most important in this dataset.

9.Saving Model – Random Forest

```
import pickle
# Firstly we will be using the dump() function to save the model
using pickle
saved_model = pickle.dumps(rfc)
# Then we will be loading that saved model
rfc_from_pickle = pickle.loads(saved_model)
```

```
# lastly, after loading that model we will use this to make
predictions
rfc_from_pickle.predict(X_test)
```

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

10.predicting a person from the 763th row data

```
prediction=rfc.predict([[10,101,76,48,180,32.9,0.171,63]])  #
763 th
if (prediction[0] == 0):
  print('The person is not diabetic')
else:
  print('The person is diabetic')
```

```
The person is not diabetic
```

---

our ai prediction system successfully developed

Conclusion

After using all these patient records, we are able to build a machine learning model (random forest – best one) to accurately predict whether or not the patients in the dataset have diabetes or not