AI_BASED DIABETES PREDECTION SYSTEM

TEAM MEMBER

922321106031:R.SHARMI.

PROJECT: Diabetes prediction system.

INTRODUCTION: In this phase, we can explore innovative techniques such as ensemble methods and deep learning architectures to improve the prediction system's accuracy and robustness.

Ensemble Methods: We employ ensemble techniques like Random Forests and Gradient Boosting to synergistically combine the predictive abilities of multiple models. By aggregating the strengths of individual algorithms, our system minimizes the risk of overfitting and enhances predictive accuracy. This ensemble approach ensures that our predictions are not only dependable but also adaptable to diverse patient profiles.

Random forest technique: The Random Forest technique is used to enhance accuracy and reliability in predicting diabetes risks and outcomes. Here's how Random Forest contributes to the effectiveness of our system:

1.Robust Feature Selection: Random Forest excels at identifying the most relevant features (attributes) from complex and multidimensional patient data. It helps in selecting the key factors contributing to diabetes risk, ensuring that our predictions are based on essential information.

2.Ensembling for Enhanced Accuracy: The Random Forest algorithm creates an ensemble of decision trees, each trained on a different subset of the data. By combining the predictions of these trees, we achieve a more robust and accurate diabetes prediction model. This ensemble approach mitigates the risk of overfitting, making our system adaptable to diverse patient populations.

3.Handling Non-Linearity: Diabetes is a multifaceted condition influenced by various factors, often with nonlinear relationships. Random Forest is particularly adept at capturing complex, nonlinear patterns within the data, ensuring that our predictions account for the intricacies of the disease.

4.Handling Missing Data: Diabetes datasets may contain missing or incomplete information. Random Forest can handle missing data gracefully, reducing the need for data imputation and ensuring that our predictions are based on the available information.

5.Interpretability: Despite its complexity, Random Forest provides a degree of interpretability. We can analyze feature important scores to understand which factors play a significant role in diabetes prediction, aiding healthcare professionals in decision-making.

Gradient boosting technique: The Gradient Boosting technique is used to elevate the accuracy and effectiveness of our predictive model. Here's how Gradient Boosting contributes to the system:

1.Improved Accuracy: Gradient Boosting is known for its ability to sequentially train decision trees, each correcting the errors of its predecessor. This iterative process leads to increasingly accurate predictions, making it well-suited for complex tasks like diabetes prediction. Our system benefits from this precision, ensuring that individuals receive reliable risk assessments.

2. Feature Importance: Gradient Boosting provides valuable insights into feature importance. By analyzing the contribution of each feature to the model's predictions, we gain a deeper understanding of the factors influencing diabetes risk. This information can guide healthcare professionals in tailoring interventions and treatment plans.

3.Robustness to Outliers: Diabetes datasets may contain outliers or noisy data points. Gradient Boosting is robust to outliers and can effectively handle such data anomalies, ensuring that our predictions remain stable and reliable.

4.Adaptability: Our system's use of Gradient Boosting allows it to adapt to changing patient data over time. This adaptability is crucial for monitoring and managing diabetes effectively, as patient conditions can evolve.

5. Ensemble of Decision Trees: Similar to Random Forest, Gradient Boosting combines the predictions of multiple decision trees. However, it does so in a different manner, focusing on correcting errors made by previous trees. This ensemble approach enhances the system's robustness and generalization capability.

Deep Learning Architectures:

Embracing the potential of neural networks, our system incorporates deep learning architectures like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). These models excel at capturing intricate patterns and temporal dependencies in patient data, making them invaluable in diabetes prediction. With deep learning, our system adapts and evolves as it processes new information, ensuring robustness in the face of dynamic health conditions.

Convolutional Neural Networks (CNNs), a deep learning architecture primarily associated with image analysis, to bring a unique perspective to diabetes prediction. Here's how CNNs contribute to the system's efficiency:

1.Image-Based Data Processing: In certain scenarios, medical images, such as retinal scans or tissue samples, can contain valuable information related to diabetes risk. CNNs are exceptionally skilled at extracting meaningful features from images, allowing us to incorporate image-based data into our predictions.

2.Feature Extraction: CNNs excel at automatically extracting hierarchical features from complex data, which is particularly beneficial when dealing with intricate patterns in diabetes-related images or datasets.

3.Early Detection: Using CNNs, we can identify anomalies that may indicate early signs of diabetes-related complications, like diabetic retinopathy. Early detection is crucial for timely intervention and prevention.

4.Multi-Modal Data Integration: Our system can seamlessly integrate diverse data types, including structured patient data, textual medical records, and medical images, enabling a comprehensive approach to diabetes prediction and management.

5.Transfer Learning: CNNs can benefit from transfer learning, where pre-trained models on large image datasets are fine-tuned for diabetes-specific tasks. This approach leverages the generalization power of CNNs while tailoring them to our specific healthcare domain.

Dataset link: https://www.kaggle.com/code/sharmisenthil/fork-of-diabetespredictor-using-decision-tree-svc/edit

Data source: A good data source for AI_Based diabetes prediction system should contain patient demographics, medical history, clinical measurements, laboratory results, lifestyle factors, medication history, genetic information, socioeconomic factors, behavioral data.

Pregnancies	Glucose	Blood Pressure	Skin thickness	Insulin	BMI	Diabetes Pedigree Function	Age	Outcome
6	148	72	35	0	33.6	0.627	50	1
1	85	66	29	0	26.6	0.351	31	0
8	183	64	0	0	23.3	0.672	32	1
1	89	66	23	94	28.1	0.167	21	0
0	137	40	35	168	43.1	2.288	33	1
5	116	74	0	0	25.6	0.201	30	0
3	78	50	32	88	31	0.248	26	1
10	115	0	0	0	35.3	0.134	29	0
2	197	70	45	543	30.5	0.158	53	1
8	125	96	0	0	0	0.232	54	1
4	110	92	0	0	37.6	0.191	30	0
10	168	74	0	0	38	0.537	34	1
10	139	80	0	0	27.1	1.441	57	0
1	189	60	23	846	30.1	0.398	59	1
5	166	72	19	175	25.8	0.587	51	1
7	100	0	0	0	30	0.484	32	1
0	118	84	47	230	45.8	0.551	31	1
7	107	74	0	0	29.6	0.254	31	1
1	103	30	38	83	43.3	0.183	33	0
1	115	70	30	96	34.6	0.529	32	1
3	126	88	41	235	39.3	0.704	27	0
8	99	84	0	0	35.4	0.388	50	0
7	196	90	0	0	39.8	0.451	41	1
9	119	80	35	0	29	0.263	29	1
11	143	94	33	146	36.6	0.254	51	1

Program: Diabetes prediction s

Importing dependencies

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

Loading dataset

```
df=pd.read_csv('https://in.docworkspace.com/d/sIDPqyObiAeKQlakG')
```

Data cleaning: Remove or handle missing values, outliers and inconsistencies in the dataset. This ensures that the data used for training is of high quality.
Handling missing data: Address missing data by imputation techniques like mean, median or using advanced methods such as K- nearest neighbors' imputation.
Data normalization: Normalize numerical features to bring them to a similar scale, which can help models' coverage faster.

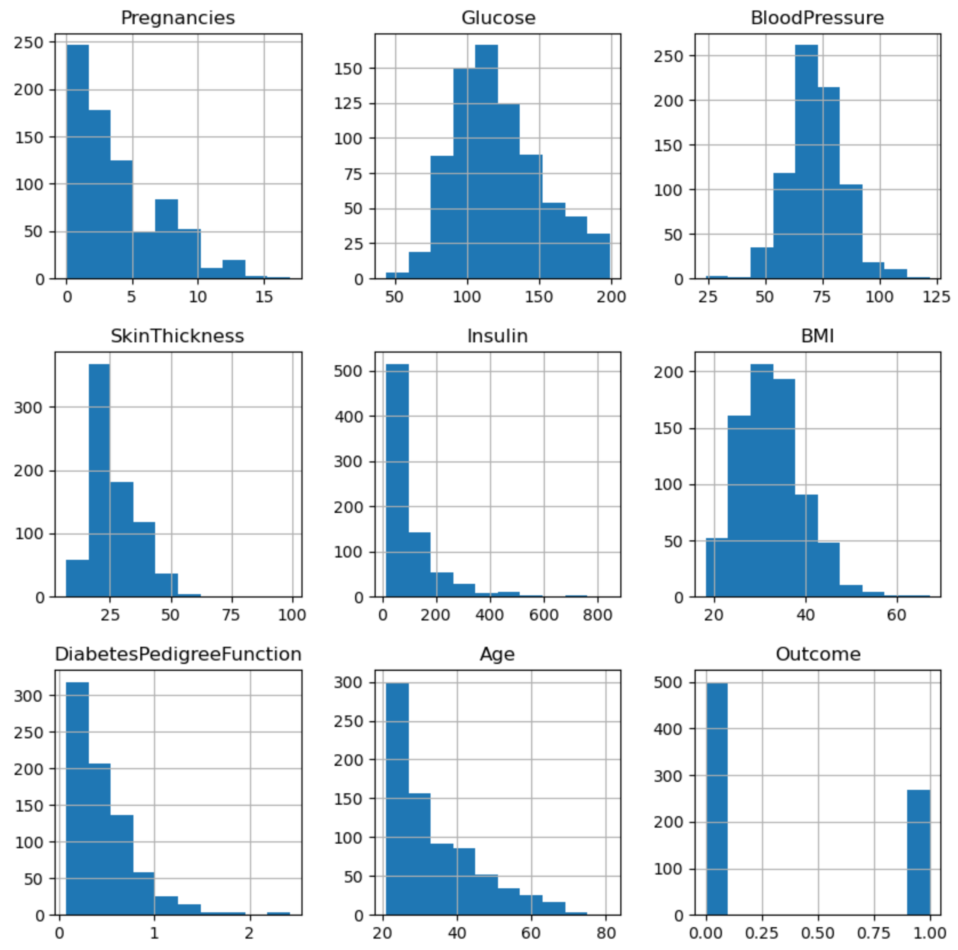
```
df=df.drop_duplicates()
```

```
df.isnull().sum()
```

```
Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

Histogram processing: It can help you visualize and understand the distribution of various features like blood glucose levels, insulin levels, BMI, etc., in your dataset. This can reveal outliers, skewness, and help you decide on data normalization or transformation techniques.

```
df.hist(bins=10,figsize=(10,10))
plt.show()
```



Model Selection: Consider various machine learning models such as logistic regression, decision trees, random forests, support vector machines, neural networks, and more.

Feature Importance: Train a machine learning model and assess feature importance scores. Models like Random Forest provide a built-in feature importance ranking, which can guide your selection process.
Hyper parameter Tuning: For selected models, fine-tune hyper parameters (e.g., learning rate, depth of trees, and number of hidden layers) using techniques like grid search or random search.

```
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test=
train_test_split(X,Y,test_size=0.2,random_state=0)
```

```
X_train.shape,Y_train.shape
```

```
((614, 5), (614,))
```

```
X_test.shape,Y_test.shape
```

```
((154, 5), (154,))
```

Decision trees: These are used as building blocks in ensemble methods like Random Forests, where multiple decision trees are combined to improve prediction accuracy and reduce overfitting.

```
import numpy as np
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score


DecisionTreeClassifier(criterion='entropy')
```

```
entropy_tree = DecisionTreeClassifier(criterion='entropy')
entropy_tree.fit(X_train, Y_train)


 Make predictions on the test set
entropy_predictions = entropy_tree.predict(X_test)
```

```
Calculate the accuracy of the model
entropy_accuracy = accuracy_score(Y_test, entropy_predictions)

Print the accuracy
print(f'Entropy Accuracy: {entropy_accuracy}')
```

```
Entropy Accuracy: 0.7272727272727273
```

```
from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier(criterion='entropy',max_depth=5)
dtc.fit(X_train, Y_train)

dtc_acc= accuracy_score(Y_test,dtc.predict(X_test))

print("Train Set
Accuracy:"+str(accuracy_score(Y_train,dtc.predict(X_train))*100))
print("Test Set
Accuracy:"+str(accuracy_score(Y_test,dtc.predict(X_test))*100))
```

```
Train Set Accuracy:80.61889250814332
Test Set Accuracy:77.92207792207793
```

```
Split the dataset into 85% training and 15% testing
from sklearn.model_selection import train_test_split
X_train , X_test , y_train , y_test =
train_test_split(X,y,test_size=0.15,random_state=42)

Scale the data (Feature Scaling)
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)

Create a function for model
def models(X_train,y_train):

 LogisticRegression

 from sklearn.linear_model import LogisticRegression
 log = Logistic Regression(solver='liblinear')
 log.fit(X_train,y_train)




Random Forest

from sklearn.ensemble import RandomForestClassifier
    rnd =
RandomForestClassifier(n_estimators=70,random_state=0,criterion='gini',max
_depth=14,bootstrap=False,verbose = 0)
    rnd.fit(X_train,y_train)

    Gradient Boost
    from sklearn.ensemble import GradientBoostingClassifier
    gb = GradientBoostingClassifier()
    gb.fit(X_train,y_train)


Print the model accuracy of training data
    print('Logistic Regression Training Accuracy : ',log.score(X_train,
y_train))
    print('Random Forest Training Accuracy : ',rnd.score(X_train,
y_train))
    print('Gradient Boosting Classifier Training Accuracy :
',gb.score(X_train, y_train))


    return log,rnd,gb

Get the model
model = models(X_train,y_train)

Logistic Regression Training Accuracy :  0.8190184049079755
Random Forest Training Accuracy :  1.0
Gradient Boosting Classifier Training Accuracy :  0.9846625766871165

Test model accuracy on test data using confusion matrix and accuracy score

from sklearn.metrics import confusion_matrix , accuracy_score
for x in range(len(model)):
    print('Model :',model[x])
    cm = confusion_matrix(y_test,model[x].predict(X_test))
    print(cm)
    print('Accuracy ',accuracy_score(y_test,model[x].predict(X_test)))
    print('\n\n')
```

```
Model : LogisticRegression(solver='liblinear')
[[61 15]
 [14 26]]
Accuracy  0.75

Model : RandomForestClassifier(bootstrap=False, max_depth=14,
n_estimators=70,random_state=0)
[[64 12]
 [14 26]]
Accuracy  0.7758620689655172

Model : GradientBoostingClassifier()
[[64 12]
 [13 27]]
Accuracy  0.78448275862068
```

```
from sklearn.metrics import confusion_matrix, accuracy_score,
precision_score
```

```
cnf_matrix = confusion_matrix(y_pred, y_test)
```

```
cnf_matrix
```

```
array([[119,  26],
       [ 11,  36]])
```

Evaluation

```
X = data.drop("Outcome", axis=1)
y = data["Outcome"]
```

Split the data into training and testing sets (70% train, 30% test)

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)
```

Create a Random Forest Classifier

```
rf_classifier = RandomForestClassifier(n_estimators=100, random_state=42)

Train the classifier on the training data
rf_classifier.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

Conclusion: By combining these techniques, you can create a deep learning-based diabetic prediction system that is not only accurate but also robust in handling diverse and evolving data.

AI based diabetes prediction systems hold

great promise for the future of healthcare by enabling early intervention, personalized healthcare and data-driven insights. However, they must be developed and deployed with careful consideration of ethical, privacy and clinical factors to ensure their positive impact on individuals and the healthcare system as a whole.

As technology and understanding in this field continue to evolve, responsible and patient-centered implementation will be key to realizing the full potential of AI in diabetes prediction and care.