Abhishek Jani **IT Department**

Roll No - 50

Experiment No.6
Predict disease risk from patient data
Date of Performance: 16/10/2023
Date of Submission:28/10/2023

Aim: To predict disease risk from patient data
Objective: Develop an artificial intelligence-based predictive model to assess the risk of Type 2 diabetes in adults using a dataset of patient demographics, medical history, and lifestyle factors, with the aim of early intervention and personalized preventive healthcare strategies.
Theory:
Predicting disease risk from patient data using artificial intelligence (AI) in healthcare involves the use of machine learning and data analysis techniques to analyze patient information and identify potential health risks. Here's a general framework for building a disease risk prediction model:

Data Collection:
Gather a comprehensive dataset that includes patient information relevant to the disease you want to predict. This data may include medical history, family history, demographics, lifestyle factors, clinical tests, and genetic information.

Data Preprocessing:
Clean and preprocess the data to handle missing values, outliers, and ensure it's in a suitable format for analysis. This may involve data normalization, feature engineering, and data encoding.

Feature Selection:

Identify which features (variables) are most relevant for predicting the disease. Feature selection helps in reducing noise in the data and improving model performance.

Data Splitting:
Split the dataset into training, validation, and test sets. The training set is used to train the AI model, the validation set is used for hyperparameter tuning, and the test set is used to evaluate the model's performance.

Model Selection:
Choose an appropriate machine learning model for disease risk prediction. Common models include logistic regression, decision trees, random forests, support vector machines, and deep learning models (e.g., neural networks).

Model Training:
Train the selected model on the training data. The model learns to recognize patterns and relationships within the data.

Hyperparameter Tuning:
Optimize the model's hyperparameters to improve its performance. Techniques like grid search or random search can be used for this purpose.

Model Evaluation:
Evaluate the model's performance using the validation set. Common evaluation metrics for classification tasks include accuracy, precision, recall, F1 score, and area under the receiver operating characteristic curve (AUC-ROC).

Testing and Validation:

Assess the model's performance on the test set, which represents new, unseen data. This step helps determine how well the model generalizes to real-world cases.

Deployment:
If the model performs well, it can be deployed in a healthcare setting, such as a hospital or clinic, where it can analyze patient data and provide risk predictions in real-time.

Continuous Monitoring and Improvement:
Regularly update the model and its data to account for changes in patient profiles, medical guidelines, and the availability of new data. Monitoring the model's performance and retraining as needed is crucial for maintaining accuracy.

Ethical Considerations and Privacy:
Be mindful of ethical considerations and patient privacy when handling medical data. Compliance with relevant regulations and obtaining patient consent is essential.

Interpretability and Explainability:
Ensure that the AI model is interpretable and can provide explanations for its predictions, especially in the healthcare domain, where transparency is critical.

The choice of the specific AI algorithms and techniques will depend on the disease being predicted, the available data, and the goals of the prediction. Also, it's important to involve healthcare professionals and domain experts throughout the process to ensure that the predictions align with clinical knowledge and best practices.

Code: -

```
[14] # Import necessary libraries
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report

# Load your dataset (you need to have a dataset with relevant features)
# For this example, let's assume you have a CSV file named "diabetes_dataset.csv"
# Replace 'your_data.csv' with your dataset file path.
data = pd.read_csv('diabetes_prediction_dataset.csv')
data
```

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	Female	80.0	0	1	never	25.19	6.6	140	0
1	Female	54.0	0	0	No Info	27.32	6.6	80	0
2	Male	28.0	0	0	never	27.32	5.7	158	0
3	Female	36.0	0	0	current	23.45	5.0	155	0
4	Male	76.0	1	1	current	20.14	4.8	155	0
...	...	...	...	...	...	...	...	...	...
99995	Female	80.0	0	0	No Info	27.32	6.2	90	0
99996	Female	2.0	0	0	No Info	17.37	6.5	100	0
99997	Male	66.0	0	0	former	27.83	5.7	155	0
99998	Female	24.0	0	0	never	35.42	4.0	100	0
99999	Female	57.0	0	0	current	22.43	6.6	90	0

Connected to Python 3 Google Compute Engine backend

```
# Assuming you have a DataFrame 'df' with a 'gender' column
data['gender'] = data['gender'].map({'Male': 0, 'Female': 1})
data['smoking_history'] = data['smoking_history'].map({'never': 0, 'No Info': 1,'current': 2,'former': 3})
data
```

	gender	age	hypertension	heart_disease	smoking_history	bmi	HbA1c_level	blood_glucose_level	diabetes
0	1.0	80.0	0	1	0.0	25.19	6.6	140	0
1	1.0	54.0	0	0	1.0	27.32	6.6	80	0
2	0.0	28.0	0	0	0.0	27.32	5.7	158	0
3	1.0	36.0	0	0	2.0	23.45	5.0	155	0
4	0.0	76.0	1	1	2.0	20.14	4.8	155	0
...	...	...	...	...	...	...	...	...	...
99995	1.0	80.0	0	0	1.0	27.32	6.2	90	0
99996	1.0	2.0	0	0	1.0	17.37	6.5	100	0
99997	0.0	66.0	0	0	3.0	27.83	5.7	155	0
99998	1.0	24.0	0	0	0.0	35.42	4.0	100	0
99999	1.0	57.0	0	0	2.0	22.43	6.6	90	0

100000 rows × 9 columns

```python
# Import necessary libraries
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.impute import SimpleImputer

# Split the data into features (X) and target (y)
X = data.drop('diabetes', axis=1)  # Replace 'Diabetes' with your target column name
y = data['diabetes']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Impute missing values
imputer = SimpleImputer(strategy='mean')
X_train = imputer.fit_transform(X_train)
X_test = imputer.transform(X_test)

# Create and train a random forest classifier (you can choose a different classifier as needed)
clf = RandomForestClassifier(n_estimators=100, random_state=42)
clf.fit(X_train, y_train)

# Make predictions on the test set
y_pred = clf.predict(X_test)

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
```

```python
# Make predictions on the test set
y_pred = clf.predict(X_test)

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Print the results
print(f'Accuracy: {accuracy}')
print(f'Classification Report:\n{classification_rep}')
```

```
Accuracy: 0.9698
Classification Report:
              precision    recall  f1-score   support

           0       0.97      1.00      0.98     18292
           1       0.94      0.69      0.80      1708

    accuracy                           0.97     20000
   macro avg       0.96      0.84      0.89     20000
weighted avg       0.97      0.97      0.97     20000
```

Google Collaboratory Link: -

AIHC-EXPT6.ipynb

Conclusion: -

Thus, we have successfully predicted disease risk from patient dataset for diabetes.