

PHASE 2 : PROJECT

PROBLEM SOLVING AND DESIGN THINKING -INNOVATION

TITLE : AI BASED DIABETES PREDICTION SYSTEMS_PHASE 2

Name : B . Abirame Susee

Roll no:202109

St.Xavier Catholic college of engineering



agenda



- ❖ Introduction
- ❖ Problem Definition
- ❖ Problem Explanation
- ❖ Design
- ❖ Machine Learning Classifiers
- ❖ Types of Machine learning classifiers
- ❖ Results and Discussion
- ❖ Algorithm for AI based prediction system using Machine learning
- ❖ Design Procedure
- ❖ Conclusion

Introduction

After using the 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 data set have diabetes or not along with that we are able to draw some insights from the data via data analysis and visualization.

Problem Definition

The problem is to build an AI powered spam classifier that can accurately distinguish between spam and non spam message in email or text messages.

The goal is to reduce the number of false positives (classifying legitimate messages as spam) and false negatives (missing actual spam messages) while achieving a high level of accuracy.

Problem Explanation

Diabetes prediction is a classification technique with mutually exclusive possible outcomes either the person is diabetic or non diabetic on careful examination of the performance of techniques used in prevalent works, logistic regression, random forest, decision tree.

Design

The following techniques are used in diabetes prediction system including,

- i. **Data set** : The dataset comprises six features that is pregnancy, glucose, blood pressure, skin thickness, BMI, age and outcome of diabetes from 203 female individuals aged between 18 and 77.
- ii. **Dataset preprocessor** : In the merged dataset preprocessing, discovered a few exceptional zero values. The zero values has been replaced by its corresponding mean value.

$$\text{RMSE} = \sqrt{\sum_{i=1}^N (\text{predicted}_i - \text{actual}_i)^2 \setminus N}$$

where, N is the total number of validation samples.

$$X_{\text{scaled}} = \frac{X - X_{\min}}{X_{\max} - X_{\min}}$$

here X_{\max} and X_{\min} denotes the maximum and minimum values in the individual feature column.

Machine Learning Classifiers

Machine learning and ensemble techniques have been employed to implement the automatic diabetes prediction system . Machine Learning are classified into following types .

Types of Machine Learning :

- **Random Forest** : Random forest is a machine learning system that averages the predictions of several decision time.
- **Logistic Regression**: Logistic regression can be used to predict a binary class .To predict the outcome , it fits an 's' shaped function.
- **Ada Boost** : Ada boost is an assemble technique . This classifier initially works on the original dataset.

- **XG Boost :** XG Boost is an ensemble machine learning technique based on decision trees that employ a gradient boosting approach.

Voting Classifier : It is an ensemble technique to improve the classification by voting. This paper implemented a voting classifier that selects the majority classifier with 'soft' voting hyper parameter.

Bagging : Bagging classifiers are ensemble classifiers that fit base classifiers to random subsets of original dataset.

Results and Discussion

This section presents the results and discussion of the proposed automatic diabetes prediction system. Equations of these metrics are expressed as ,

$$\text{Precision} = \frac{T_p}{T_p + F_p}$$

$$\text{Recall} = \frac{T_p}{T_p + F_N}$$

where , “ T_p ” denotes model is predicting positive and the result is also positive.

“ F_p ” indicates the positive prediction of the model, but the result is negative.

“ T_N ” expresses the model is predicting negative and the result is positive.

Program

```
Importing the necessary librariesimport pandas as pdfrom
sklearn.model_selection import train_test_splitfrom
sklearn.preprocessing import StandardScalerfrom sklearn.svm import
SVC# Loading the datasetdiabetes_data = pd.read_csv("diabetes.csv")#
Splitting the dataset into features and target variableX =
diabetes_data.drop("Outcome", axis=1)y = diabetes_data["Outcome"]#
Splitting the dataset into training and testing setsX_train, X_test, y_train,
y_test = train_test_split(X, y, test_size=0.2, random_state=42)# Scaling the
featurescaler = StandardScaler()X_train =
scaler.fit_transform(X_train)X_test = scaler.transform(X_test)# Creating
and training the SVM classifierclassifier = SVC()classifier.fit(X_train,
y_train)# Making predictions on the test setpredictions =
classifier.predict(X_test)# Evaluating the modelaccuracy =
classifier.score(X_test, y_test)print("Accuracy:", accuracy)
```

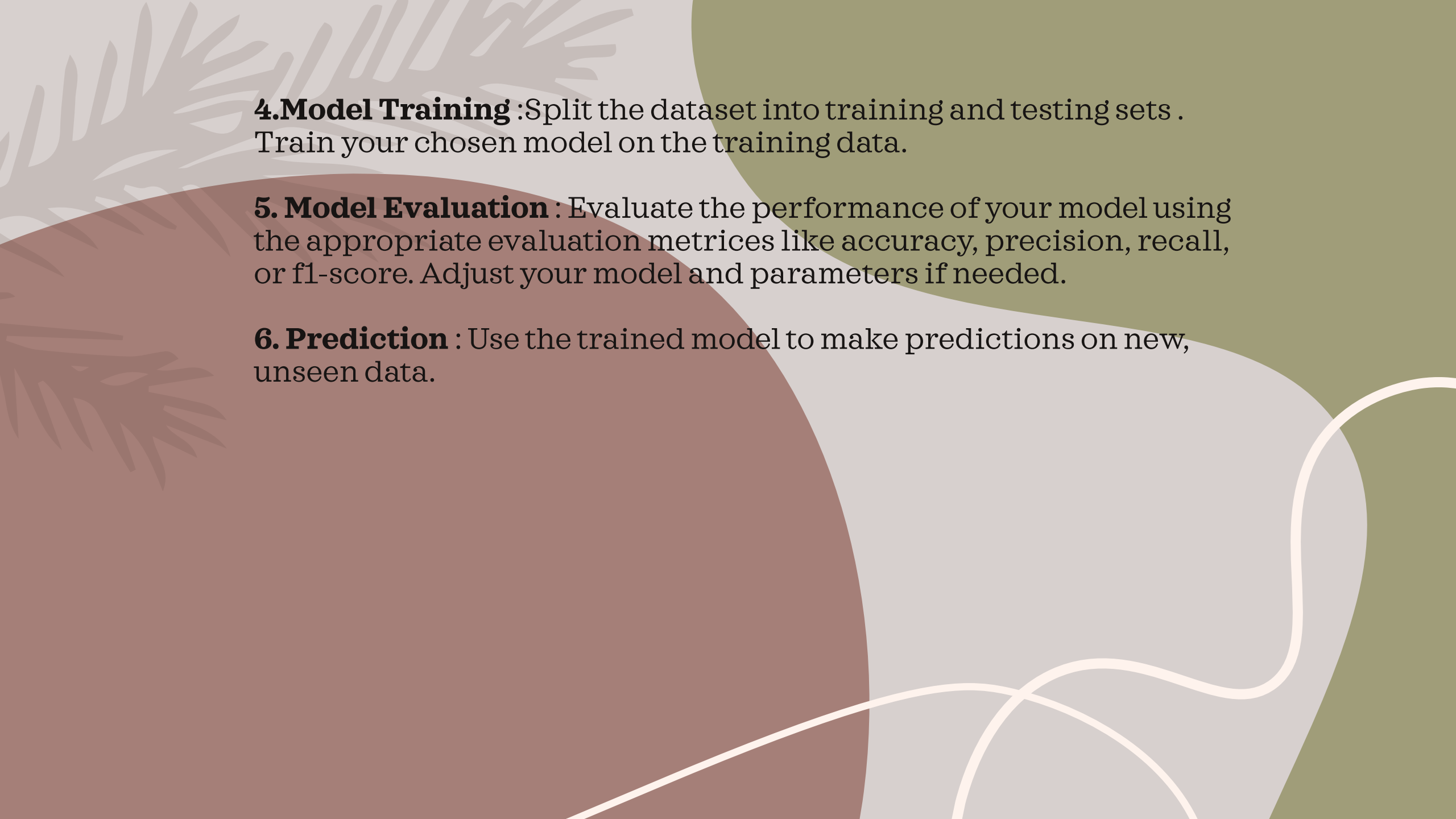
Design Procedure

To design an AI based diabetes prediction system using the Pima Indians diabetes database, can follow these general steps:

1. Data preprocessing : clean and preprocess the dataset by handling missing values, normalizing features, and handling outliers if necessary.

2. Feature selection : identify the most relevant features that can contribute to diabetes prediction .You can use techniques like correlation analysis of feature importance ranking.

3.Model selection : choose an appropriate machine learning algorithm for your prediction task , such as logistic regression, decision trees, random forests, or support vector machines.



4. Model Training : Split the dataset into training and testing sets .
Train your chosen model on the training data.

5. Model Evaluation : Evaluate the performance of your model using the appropriate evaluation metrics like accuracy, precision, recall, or f1-score. Adjust your model and parameters if needed.

6. Prediction : Use the trained model to make predictions on new, unseen data.

Dataset Programs

In [1]:

```
#Let's start with importing necessary l
ibraries
import pandas as pd
import numpy as np
from sklearn.preprocessing import Stan
dardScaler
from sklearn.linear_model import Logi
sticRegression
from sklearn.model_selection import tr
ain_test_split
from sklearn.metrics import accuracy_s
core, confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns
```

In [12]:

X_train_scaled

Out[12]:

```
array([[ 1.50755225, -1.09947934, -0.8
9942504, ..., -1.45561965,
        -0.98325882, -0.04863985],
       [-0.82986389, -0.1331471 , -1.2
3618124, ...,  0.09272955,
        -0.62493647, -0.88246592],
       [-1.12204091, -1.03283573,  0.6
1597784, ..., -0.03629955,
        0.39884168, -0.5489355 ],
       ...,
       [ 0.04666716, -0.93287033, -0.6
4685789, ..., -1.14021518,
        -0.96519215, -1.04923114],
       [ 2.09190629, -1.23276654,  0.1
1084355, ..., -0.36604058,
        -0.5075031 ,  0.11812536],
       [ 0.33884418,  0.46664532,  0.7
8435594, ..., -0.09470985,
        0.51627505,  2.953134  ]])
```

In [11]:

```
X_train_scaled, X_test_scaled = scaler
_standard(X_train, X_test)
```

In [12]:

X_train_scaled

In [11]:

```
X_train_scaled, X_test_scaled = scaler
_standard(X_train, X_test)
```

In [12]:

X_train_scaled

In [10]:

```
import pickle
##standard Scaling- Standardization
def scaler_standard(X_train, X_test):
    #scaling the data
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    #saving the model
    file = open('standardScalar.pkl', 'wb')
    pickle.dump(scaler, file)
    file.close()

    return X_train_scaled, X_test_scaled
```

In [10]:

```
import pickle
##standard Scaling- Standardization
def scaler_standard(X_train, X_test):
    #scaling the data
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    #saving the model
    file = open('standardScalar.pkl', 'wb')
    pickle.dump(scaler, file)
    file.close()

    return X_train_scaled, X_test_scaled
```

In [5]:

```
#here few misconception is there like BMI can not be zero, BP can't be zero, glucose, insulin can't be zero so let's try to fix it
# now replacing zero values with the mean of the column
data['BMI'] = data['BMI'].replace(0, data['BMI'].mean())
data['BloodPressure'] = data['BloodPressure'].replace(0, data['BloodPressure'].mean())
data['Glucose'] = data['Glucose'].replace(0, data['Glucose'].mean())
data['Insulin'] = data['Insulin'].replace(0, data['Insulin'].mean())
data['SkinThickness'] = data['SkinThickness'].replace(0, data['SkinThickness'].mean())
```

In [6]:

```
#now we have dealt with the 0 values and data looks better. But, there still are outliers present in some columns. let's visualize it
fig, ax = plt.subplots(figsize=(15, 10))
sns.boxplot(data=data, width=0.5, ax=ax, fliersize=3)
```

Algorithm to Create a Chatbot in Python:

Step 1: Install NLTK using pip:

Pip install nltk

Step 2: Import the necessary libraries:

Import nltk

From nltk.chat.util import Chat

Step 3: Create a list of patterns and responses. Each pattern-response pair should be a tuple.


```
pairs = [ [ r"hi | hello | hey",  
["Hello!", "how are you",  
"How can I help you ?"] ],  
[["I'm good, thanks. How about you?", "I'm a chatbot, so I'm always fine."]],]
```

Step 4: Initialize the chatbot using the Chat class from NLTK:

```
chatbot = Chat (pairs, reflections)
```

Step 5: Create a function to start the chat and interact with the user:

```
def chat_with_user(): print("Hello! I'm your chatbot. Type 'exit'  
to end the conversation.")
```

Step 6: Call the chat_with_bot() function to start the bot

Conclusion

The objective of the project was to develop a model with which could identify patients with diabetes who are high risk of the disease . The model makes the prediction with an accuracy of 98%.