AI Based Diabetics Prediction System Project Report

Team Members: 5

Team Leader: THANGESWARAN S

TEAM MEMBERS:

1. VINOTHKUMAR.R

2. SHREEKANTH.M

3. SRINIVASAN.M

4. THANGESWARAN.S

5. VISHAL.N

INDEX :

Problem Statement:

- Develop an AI-based system for predicting the risk of diabetes in individuals. The system should analyze various health-related data and provide accurate predictions to enable early intervention and proactive healthcare management for at-risk individuals.

Design Thinking Process:

Empathize:

- Understand the perspective of potential users, including patients, healthcare providers, and researchers. Gather insights on their needs, challenges, and expectations regarding diabetes prediction and prevention.

Define:

- Define the problem based on the insights gained from empathizing. Formulate a clear problem statement that addresses the key pain points and requirements of the users.

Ideate:

- Generate a wide range of ideas and possible solutions for building an AI-based diabetes prediction system. Encourage creativity and explore various approaches that could address the problem effectively.

Prototype:

- Develop a prototype of the AI system. This could involve creating a basic model that uses machine learning algorithms to analyze health data and predict the risk of diabetes. The prototype should be flexible for iterative improvements based on feedback.

Test:

- Test the prototype with a sample dataset to evaluate its accuracy, usability, and reliability in predicting diabetes risk. Gather feedback from users and experts to identify strengths and weaknesses.

Develop:

- Refine the system based on the feedback received during testing. Enhance the algorithm, improve the user interface, and ensure the system is scalable and adaptable to different datasets and user needs.

Implement:

- Deploy the developed AI-based diabetes prediction system in a controlled environment. Monitor its performance, gather real-world data, and continue refining the system based on user feedback and evolving needs.

Evaluate:

- Regularly evaluate the system's performance, accuracy, and user satisfaction. Collect real-world data to continuously improve the system's predictive capabilities.

Phases of Development: **Research and Data Collection**:

- Gather relevant health data, including medical records, lifestyle information, genetic predispositions, and other factors associated with diabetes.

Data Preprocessing:

- Clean and preprocess the collected data to ensure its quality and relevance for the AI system. This involves handling missing values, normalizing data, and preparing it for analysis.

Feature Selection and Engineering:

- Identify the most relevant features that contribute to diabetes prediction. This step involves selecting the right data attributes and potentially creating new features that might enhance prediction accuracy.

Model Development:

- Utilize machine learning algorithms (such as logistic regression, decision trees, random forests, neural networks, etc.) to build a predictive model. Train the model using labeled data to predict the likelihood of diabetes.

Model Evaluation:

- Assess the model's performance using metrics like accuracy, precision, recall, and F1 score. Validate the model's predictions using separate test datasets to ensure its reliability.

User Interface Development:

- Design an intuitive user interface for the system, allowing easy input of data and displaying predictions in a user-friendly manner.

Integration and Deployment:

- Integrate the developed model and user interface into a cohesive system. Deploy the system in a controlled environment for testing and validation.

Feedback and Iteration:

- Gather feedback from users, healthcare professionals, and other stakeholders. Use this feedback to make necessary iterations, improve the model's accuracy, and enhance the system's usability.

Continuous Improvement:

- Continuously update and improve the system by incorporating new data, refining the model, and updating the user interface based on ongoing feedback and advancements in technology.

Source Code

```
# Source code for datahandling using python
import pandas as pd import numpy as np
import matplotlib.pyplot as plt import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix, roc_curve, auc from sklearn.preprocessing import StandardScaler

# Load the dataset
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'] data = pd.read_csv(url, names=names)

# Split the dataset into features (X) and target (y) X = data.drop('Outcome', axis=1)
y = data['Outcome']

# 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)

# Feature scaling (Standardization) scaler = StandardScaler()
X_train = scaler.fit_transform(X_train) X_test = scaler.transform(X_test)

# Create and tune a Random Forest Classifier param_grid = {
```

```python
    'n_estimators': [50, 100, 200],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4]
}

rf = RandomForestClassifier(random_state=42)
grid_search = GridSearchCV(estimator=rf, param_grid=param_grid, cv=5, n_jobs=-
1) grid_search.fit(X_train, y_train)
best_rf = grid_search.best_estimator_

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

# Evaluate the model's performance accuracy = accuracy_score(y_test, y_pred) print(f"Accuracy: {accura
cy:.2f}")

# Classification Report print("\nClassification Report:") print(classification_report(y_test, y_pred))

# Confusion Matrix print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# ROC curve and AUC
y_prob = best_rf.predict_proba(X_test)[:, 1] fpr, tpr, thresholds = roc_curve(y_test, y_prob) roc_auc = au
c(fpr, tpr)

# Data Visualization plt.figure(figsize=(10, 6))
plt.subplot(1, 2, 1)
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt="d", cmap="Blues") plt.title("Confusion
 Matrix")

plt.subplot(1, 2, 2)
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})') plt.plot([0, 1], [0, 1],
color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05]) plt.xlabel('False Positive Rate') plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)') plt.legend(loc='lower right')

plt.tight_layout() plt.show()
```

OUTPUT: Accuracy: 0.75

Classification Report:

precision
recall
f1-score
support 0 0.81 0.80 0.80 99 1 0.64 0.65 0.65 55 accuracy 0.75 154 macro avg 0.72 0.73 0.73 154
weighted avg 0.75 0.75 0.75 154

Confusion Matrix: [[79 20]
[19 36]]

- **Selecting a machine learning algorithm**

- **training the model**

- **Evaluating its performance**

- As Per the Mentor of IBM show above we have developed Our Project Using python
  Programming language and steps and Source code andOutput is give below to Evaluate...Steps
  By Step Involved In Building Project Shown Below:

Step 1: *In this code, we use the Pima Indian Diabetes dataset, split it into training and testing sets, and
train a Random Forest Classifier. We then evaluate the model's accuracy.

Step 2:

- Import Necessary Libraries: Open a Python environment or Jupyter Notebook and start by
  importing the required libraries:

```
# This is formatted as code
from sklearn.model_selection
import train_test_split from sklearn.ensemble
import RandomForestClassifier from sklearn.metrics
import accuracy_score
```

Step 3:

- Load the Dataset: Download the Pima Indian Diabetes dataset (or) any source of dataset can be
  utilized from this link and save it as aCSV file. Then, load the dataset into your script: We have
  utilized a github public repository as free of cost in our project as show below code
  representation.

```
# This is formatted as code
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome'] data = pd.read_csv(url, names=names)
```

Step 4:

- Split Data into Features and Target: Separate the dataset into features (X) and the target
  variable (y):

Step 5:

- Split Data into Training and Testing Sets:

This is formatted as code
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Step 6:

- : Creating (or) Utilizing the Algorithm a Random Forest Classifier and train it on the training data:

# This is formatted as code
model = RandomForestClassifier() model.fit(X_train, y_train)

Step 7:

- : Make Predictions and Evaluate the Model of Trained Dataset using Random Forest Classifier Algorithm:

This is formatted as code

accuracy = accuracy_score(y_test, y_pred) print(f"Accuracy: {accuracy:.2f}")
# Classification Report print("\nClassification Report:") print(classification_report(y_test, y_pred))

# Confusion Matrix print("\nConfusion Matrix:")
print(confusion_matrix(y_test, y_pred))

# ROC curve and AUC
y_prob = best_rf.predict_proba(X_test)[:, 1] fpr, tpr, thresholds = roc_curve(y_test, y_prob) roc_auc = auc(fpr, tpr)

# Data Visualization plt.figure(figsize=(10, 6))
plt.subplot(1, 2, 1)
sns.heatmap(confusion_matrix(y_test, y_pred), annot=True, fmt="d", cmap="Blues") plt.title("Confusion Matrix")

plt.subplot(1, 2, 2)
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (area = {roc_auc:.2f})') plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05]) plt.xlabel('False Positive Rate') plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC)') plt.legend(loc='lower right')

plt.tight_layout() plt.show()

- **Now We Have Developed a Project Using python and Machine Learning as Desktop Application To Demonstrate the Project**:

Source Code:

```
# Source Code
# -*- coding: utf-8 -*-

# Form implementation generated from reading ui file 'code.ui'
#
# Created by: PyQt5 UI code generator 5.15.9
#
# WARNING: Any manual changes made to this file will be lost when pyuic5 is
# run again.  Do not edit this file unless you know what you are doing.


from PyQt5 import QtCore, QtGui, QtWidgets
import numpy as np
import keras

class Ui_Form(object):
    def setupUi(self, Form):
        Form.setObjectName("Form")
        Form.resize(787, 661)
        Form.setMouseTracking(True)
        Form.setContextMenuPolicy(QtCore.Qt.DefaultContextMenu)
        Form.setAcceptDrops(False)
        Form.setStyleSheet("color: rgb(255, 255, 255);")
        self.labelTitre = QtWidgets.QLabel(Form)
        self.labelTitre.setEnabled(True)
        self.labelTitre.setGeometry(QtCore.QRect(140, 60, 511, 41))
        font = QtGui.QFont()
        font.setFamily("MS Shell Dlg 2")
        font.setPointSize(13)
        font.setBold(True)
        font.setItalic(False)
        font.setWeight(75)
        self.labelTitre.setFont(font)
        self.labelTitre.setCursor(QtGui.QCursor(QtCore.Qt.ArrowCursor))
        self.labelTitre.setMouseTracking(True)
        self.labelTitre.setAutoFillBackground(False)
        self.labelTitre.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
        self.labelTitre.setScaledContents(False)
        self.labelTitre.setObjectName("labelTitre")
        self.lineEditNpreg = QtWidgets.QLineEdit(Form)
        self.lineEditNpreg.setGeometry(QtCore.QRect(320, 190, 231, 31))
```

```python
        self.lineEditNpreg.setTabletTracking(True)
        self.lineEditNpreg.setContextMenuPolicy(QtCore.Qt.DefaultContextMenu)
        self.lineEditNpreg.setAcceptDrops(True)
        self.lineEditNpreg.setWhatsThis("")
        self.lineEditNpreg.setPlaceholderText("0 if male")
        self.lineEditNpreg.setAccessibleName("")
        self.lineEditNpreg.setAccessibleDescription("")
        self.lineEditNpreg.setAutoFillBackground(False)
        self.lineEditNpreg.setStyleSheet("color: rgb(0, 0, 0);")
        self.lineEditNpreg.setObjectName("lineEditNpreg")
        self.lineEditBMI = QtWidgets.QLineEdit(Form)
        self.lineEditBMI.setGeometry(QtCore.QRect(320, 240, 231, 31))
        self.lineEditBMI.setContextMenuPolicy(QtCore.Qt.DefaultContextMenu)
        self.lineEditBMI.setAcceptDrops(True)
        self.lineEditBMI.setWhatsThis("")
        self.lineEditBMI.setAccessibleName("")
        self.lineEditBMI.setPlaceholderText("0-81")
        self.lineEditBMI.setAccessibleDescription("")
        self.lineEditBMI.setStyleSheet("color: rgb(0, 0, 0);")
        self.lineEditBMI.setObjectName("lineEditBMI")
        self.lineEditGlucose = QtWidgets.QLineEdit(Form)
        self.lineEditGlucose.setGeometry(QtCore.QRect(320, 300, 231, 31))
        self.lineEditGlucose.setContextMenuPolicy(QtCore.Qt.DefaultContextMenu)
        self.lineEditGlucose.setAcceptDrops(True)
        self.lineEditGlucose.setWhatsThis("")
        self.lineEditGlucose.setPlaceholderText("0-199 mg/dl")
        self.lineEditGlucose.setAccessibleName("")
        self.lineEditGlucose.setAccessibleDescription("")
        self.lineEditGlucose.setStyleSheet("color: rgb(0, 0, 0);")
        self.lineEditGlucose.setObjectName("lineEditGlucose")
        self.lineEditAge = QtWidgets.QLineEdit(Form)
        self.lineEditAge.setGeometry(QtCore.QRect(320, 350, 231, 31))
        self.lineEditAge.setContextMenuPolicy(QtCore.Qt.DefaultContextMenu)
        self.lineEditAge.setAcceptDrops(True)
        self.lineEditAge.setWhatsThis("")
        self.lineEditAge.setPlaceholderText("1-100 is recommended")
        self.lineEditAge.setAccessibleName("")
        self.lineEditAge.setAccessibleDescription("")
        self.lineEditAge.setStyleSheet("color: rgb(0, 0, 0);")
        self.lineEditAge.setObjectName("lineEditAge")
        self.labelNpreg = QtWidgets.QLabel(Form)
        self.labelNpreg.setGeometry(QtCore.QRect(200, 200, 81, 21))
        self.labelNpreg.setStyleSheet("color: rgb(255, 255, 255);\n"
"color: rgb(0, 0, 0);\n"
```

```python
"font: 75 8pt \"MS Shell Dlg 2\";\n"
"font: 75 8pt \"MS Shell Dlg 2\";")
        self.labelNpreg.setObjectName("labelNpreg")
        self.labelGlucose = QtWidgets.QLabel(Form)
        self.labelGlucose.setGeometry(QtCore.QRect(190, 300, 101, 21))
        self.labelGlucose.setStyleSheet("color: rgb(255, 255, 255);\n"
"color: rgb(0, 0, 0);")
        self.labelGlucose.setObjectName("labelGlucose")
        self.labelBMI = QtWidgets.QLabel(Form)
        self.labelBMI.setGeometry(QtCore.QRect(210, 250, 71, 21))
        self.labelBMI.setStyleSheet("color: rgb(255, 255, 255);\n"
"color: rgb(0, 0, 0);\n"
"font: 75 8pt \"MS Shell Dlg 2\";")
        self.labelBMI.setObjectName("labelBMI")
        self.labelAge = QtWidgets.QLabel(Form)
        self.labelAge.setGeometry(QtCore.QRect(250, 360, 41, 21))
        self.labelAge.setStyleSheet("color: rgb(255, 255, 255);\n"
"color: rgb(0, 0, 0);")
        self.labelAge.setObjectName("labelAge")
        self.pushButtonClear = QtWidgets.QPushButton(Form)
        self.pushButtonClear.setGeometry(QtCore.QRect(250, 440, 111, 41))
        self.pushButtonClear.setStyleSheet("color: rgb(0, 0, 0);")
        self.pushButtonClear.setObjectName("pushButtonClear")
        self.pushButtonClear.clicked.connect(self.clear)
        self.pushButtonResult = QtWidgets.QPushButton(Form)
        self.pushButtonResult.setGeometry(QtCore.QRect(440, 440, 111, 41))
        self.pushButtonResult.setStyleSheet("color: rgb(255, 255, 255);\n"
"color: rgb(0, 0, 0);")
        self.pushButtonResult.clicked.connect(self.calculation)
        self.pushButtonResult.setObjectName("pushButtonResult")
        self.labelResults = QtWidgets.QLabel(Form)
        self.labelResults.setGeometry(QtCore.QRect(210, 540, 231, 41))
        self.labelResults.setStyleSheet("color: rgb(255, 255, 255);\n"
"color: rgb(255, 0, 0);")
        self.labelResults.setLocale(QtCore.QLocale(QtCore.QLocale.English, QtCore.QLocale.UnitedStates))
        self.labelResults.setObjectName("labelResults")
        self.lcdNumberReslults = QtWidgets.QLCDNumber(Form)
        self.lcdNumberReslults.setGeometry(QtCore.QRect(470, 540, 91, 31))
        self.lcdNumberReslults.setObjectName("lcdNumberReslults")
        self.lcdNumberReslults.setStyleSheet("color: rgb(255, 0, 0);\n"
"color: rgb(255, 0, 0);\n"
"background-color: rgb(255, 0, 0);")
        self.lcdNumberReslults.setObjectName("lcdNumberReslults")
        self.label100 = QtWidgets.QLabel(Form)
```

```python
        self.label100.setGeometry(QtCore.QRect(570, 540, 21, 31))
        self.label100.setObjectName("label100")

        self.retranslateUi(Form)
        QtCore.QMetaObject.connectSlotsByName(Form)

    def retranslateUi(self, Form):
        _translate = QtCore.QCoreApplication.translate
        Form.setWindowTitle(_translate("Form", "AI Diabetics Prediction Application"))
        self.labelTitre.setText(_translate("Form", "<html><head/><body><p align=\"justify\"><span style=\"
 font-size:14pt; text-
decoration: underline; color:#ff0000;\">AI BASED DIABETICS PREDICTION SYSTEM</span></p></body><
/html>"))
        self.labelNpreg.setText(_translate("Form", "PREGNENCIES:"))
        self.labelGlucose.setText(_translate("Form", "GLUCOSE LEVEL:"))
        self.labelBMI.setText(_translate("Form", "BMI LEVEL :"))
        self.labelAge.setText(_translate("Form", "AGE :"))
        self.pushButtonClear.setText(_translate("Form", "Clear"))
        self.pushButtonResult.setText(_translate("Form", "Result"))
        self.labelResults.setText(_translate("Form", "<html><head/><body><p><span style=\" font-
size:12pt; font-weight:600; color:#ff0000;\">GENERATED REPORT:</span></p></body></html>"))
        self.label100.setText(_translate("Form", "<html><head/><body><p><span style=\" font-
size:14pt; color:#ff0000;\">%</span></p></body></html>"))

    def clear(self, Form):
        self.lineEditNpreg.clear()
        self.lineEditBMI.clear()
        self.lineEditGlucose.clear()
        self.lineEditAge.clear()
        self.lcdNumberReslults.display(0.000)

    def results(self, Form):

        pass
    def calculation(self):
        #feature_names ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'Diabetes
PedigreeFunction', 'Age']
        model = keras.models.load_model('model_ANN.h5')
        arr    =[self.lineEditNpreg,self.lineEditBMI,self.lineEditGlucose,self.lineEditAge]
        arr ,ee = self.empty_entry_validation(arr)
        if ee == 0:
            arr    =list(map(float,arr))
            # re = self.range_validation(arr)
            # if re == 0:
```

```python
            self.arr=arr
            arr    =np.array([np.array(arr)])
            # arr    =self.model.std_scaling(arr)
            self.pred   =model.predict(arr.reshape(1,4))
            self.lcdNumberReslults.display(float(self.pred[0][0]*100))
            print(self.pred)
            # else:
            #     self.showdialog("invalid values as input")
        else:
            self.showdialog("please  enter the values to continue....")



    def empty_entry_validation(self,arr):
        l=[]
        empty=0
        for x in arr:
            l.append(x.text())
        if '' in l :
            empty=1
        for i in range(len(l)):
            if l[i]=='.':
                arr[i].clear()
                empty=1
        return l,empty



    def showdialog(self,s):
        msg = QtWidgets.QMessageBox()
        msg.setIcon(QtWidgets.QMessageBox.Information)
        msg.setText("Message Alert!")
        msg.setInformativeText("Additional information")
        msg.setWindowTitle("Hi there!")
        msg.setDetailedText(s)
        msg.setStandardButtons(QtWidgets.QMessageBox.Ok | QtWidgets.QMessageBox.Cancel)
        msg.exec_()


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    Form = QtWidgets.QWidget()
    ui = Ui_Form()
    ui.setupUi(Form)
```

```
Form.show()
sys.exit(app.exec_())
```

OUTPUT:

Requirements for Compiling the Source code: **These Library Files Has to Install Using Pip in Python**

- Seaborn

- Numpy

- Pandas

- Scikit_learn

- Tensorflow

- pyqt5

Required Source Files Are Given in github Click Here

- Explained Full Project in Github Readme file and required Files are provided in github..

Figure 1: