Course Project

1. Write a small (One Page) report about Streamlit Python Library showing its advantages and uses. Use your own language and understanding, do not Copy and Paste from AI generative tools.
2. Write Python code that does the following:
   1. Use Scikit Learn to build a simple machine learning model that does predictions e.g., Regression or Classification.
   2. Use any dataset you find suitable; you can find sample datasets and sample code on Kaggle ([Kaggle: Your Machine Learning and Data Science Community](https://www.kaggle.com/)) website.
   3. You can also find sample use cases on SharePoint ([BI-ML Use Cases](https://uopstdedu.sharepoint.com/:f:/s/BIDA/EkhozIxUfwtGm1Shk3MeBr8BkI0TLC9r3fHN3ywWpMDZfQ?e=QYLmcq))
   4. Train the model and evaluate its accuracy.
   5. Use Streamlit library to design a simple user entry form.
   6. The user entry form should include the independent variables used to train your predictive model.
   7. The user entry form should also include a button that calls your predictive model and passes the inputs provided by the user.
   8. The result of the prediction should be presented on the screen.

You can find below a sample application that demonstrates the requirements. You should design something similar using another dataset and another type of a problem.

To run and test the example file below do the following steps:

1. Install python on your machine.
2. Open command line and install Streamlit library using the following command:

*pip install streamlit*

1. In the command line, go to your project folder and type:

*Python -m streamlit run YOURFILENAME.py*

import streamlit as st

import pandas as pd

from sklearn.ensemble import RandomForestClassifier

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

# Load the diabetes dataset

diabetes\_data = pd.read\_csv("diabetes.csv")

# Split the data into features and target variable

X = diabetes\_data.drop("Outcome", axis=1)

y = diabetes\_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)

print(X\_train.info())

# Train a random forest classifier

clf = RandomForestClassifier()

clf.fit(X\_train, y\_train)

# Define the Streamlit app

def app():

    # Set the app title

    st.title("Diabetes Prediction App")

    # Add five input fields for integer values

    st.header("Enter the following details:")

    pregnancies = st.number\_input("Number of Pregnancies", min\_value=0, step=1)

    glucose = st.number\_input("Glucose Level", min\_value=0, step=1)

    blood\_pressure = st.number\_input("Blood Pressure (mm Hg)", min\_value=0, step=1)

    skin\_thickness = st.number\_input("Skin Thickness (mm)", min\_value=0, step=1)

    insulin = st.number\_input("Insulin Level (mu U/ml)", min\_value=0, step=1)

    bmi = st.number\_input("Body Mass Index BMI", min\_value=0, step=1)

    DiabetesPedigreeFunction = st.number\_input("Diabetes Pedigree Function", min\_value=0, step=1)

    age = st.number\_input("age", min\_value=0, step=1)

    # Create a button to make predictions

    if st.button("Predict"):

        # Create a DataFrame with the user input

        user\_input = pd.DataFrame(

            {

                "Pregnancies": [pregnancies],

                "Glucose": [glucose],

                "BloodPressure": [blood\_pressure],

                "SkinThickness": [skin\_thickness],

                "Insulin": [insulin],

                "BMI": [bmi],

                "DiabetesPedigreeFunction":[DiabetesPedigreeFunction],

                "Age":[age]

            }

        )

        # Make predictions using the trained classifier

        predictions = clf.predict(user\_input)

        # Display the prediction result

        st.subheader("Prediction Result:")

        if predictions[0] == 0:

            st.write("The person is not diabetic.")

        else:

            st.write("The person is diabetic.")

# Run the app

if \_\_name\_\_ == "\_\_main\_\_":

    app()

A screenshot of a medical test

Description automatically generated with low confidence