---

🎯 **Main Goals of the Project**

**1. Real-World Health Problem Understanding**

- Predict whether a person is likely to have **diabetes** based on health-related data like:
    - Blood sugar levels
    - BMI
    - Insulin levels
    - Age
    - Pregnancy count
- Helps simulate how **AI models are used in medical diagnostics** in hospitals or health tech apps.

---

**2. Learn and Practice ML Workflow with TensorFlow**

You will experience the **complete ML pipeline**:

| Phase | Goal |
|---|---|
| 📥 **Data Collection** | Load real-world health dataset (Pima Indians) |
| 🔍 **Data Analysis** | Use Pandas, Seaborn to understand the data |
| 🖌️ **Preprocessing** | Normalize features using StandardScaler |
| 🧠 **Model Building** | Build a neural network with TensorFlow |
| 🎯 **Model Training** | Train using real data to find patterns |
| ✅ **Evaluation** | Test model accuracy and compare predictions |
| 📈 **Visualization** | Plot loss curves, correlation heatmap |

---

**3. Apply Deep Learning to a Classification Task**

- This is a **binary classification** problem:
    - Outcome = 1: Person has diabetes
    - Outcome = 0: Person does not have diabetes
- Goal: **Minimize error** and correctly classify new patients

---

**4. Model Interpretation and Insight**

By the end of the project, you'll be able to:

- Read model accuracy and loss
- Understand how changing features affects predictions
- Think about feature importance in healthcare diagnostics

---

🔍 **Bonus Learning Objectives**

| Skill | Description |
|---|---|
| 💡 **Model design** | Learn structure: Input → Hidden Layers → Output |
| 📊 **Interpret loss/accuracy** | Understand performance with validation loss |
| 📈 **Visualization skills** | Build useful plots to explain findings |
| 💬 **Explainability** | Learn how to present your model in a healthcare context |

---

🧠 **Real-World Use Cases**

- 👩‍⚕️ Used by **doctors** to assist in screening patients
- 📱 Integrated in **mobile health apps** like Apple Health, Practo, etc.
- 🏦 Helps **insurance companies** assess risk of applicants

---

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns #multiple var. in ur dataset ,statistical high level ,quick

import tensorflow as tf #dev. by google , deep learning,intelligent system

from sklearn.model_selection import train_test_split

from sklearn.preprocessing import StandardScaler

#url

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/refs/heads/master/pima-indians-diabetes.data.csv"

columns = ['Pregnancies','Glucose','BloodPressure','SkinThickness','Insulin','BMI','DiabetesPedigreeFunction','Age','Outcome']
```

```
df =pd.read_csv(url,header =None,names=columns)
print(df.head())
print(df.shape)
print(df.describe())
print(df.info())

#heatmap
plt.figure(figsize=(10,6))
sns.heatmap(df.corr(),annot=True,cmap='coolwarm')
plt.title('Diabetes Outcome Distribution')
plt.show()
#diabetes  count
sns.countplot(x='Outcome',data=df)
plt.title('Count of Outcome')
plt.xlabel('Outcome')
plt.ylabel('Count')
plt.show()

#prepare the data
X = df.drop('Outcome',axis=1) #ip
y = df['Outcome'] #op
print(X.head())
print(y.head())

#scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X) #mean = 0 , sd =1
print(X_scaled)

#split the data
```

```
X_train,X_test,y_train,y_test = train_test_split(X_scaled,y,test_size=0.2,random_state=42) # 20% testing data 80% training data

#build model (fully connected (layer ip , hidden layer,output = perceptron)
model = tf.keras.Sequential([
  tf.keras.layers.Dense(64,activation='relu',input_shape=(X_train.shape[1],)),
  tf.keras.layers.Dense(32,activation='relu'),
  tf.keras.layers.Dense(1,activation='sigmoid')
])
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])

# Assign the return value of model.fit to the history variable
history = model.fit(X_train,y_train,epochs=100,batch_size=32,validation_data=(X_test,y_test))

#evaluate the model
loss,accuracy = model.evaluate(X_test,y_test)
print(f'Test Loss:{loss:.4f}')
print(f'Test Accuracy:{accuracy:.4f}')

# Print the keys in the history object to identify the correct key for accuracy
print(history.history.keys())

# Use the correct key for accuracy, which is likely 'accuracy' based on the metrics=['accuracy']
# If the print statement above shows a different key (e.g., 'acc'), replace 'accuracy' with that key
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train','Validation'],loc='upper left')
plt.show()
```

```
y_pred = model.predict(X_test)
print('\n Predictions vs Actual:')
for i in range(10):
    print(f'Predicted:{y_pred[i][0]:.2f},Actual:{y_test.iloc[i]}')
```

---