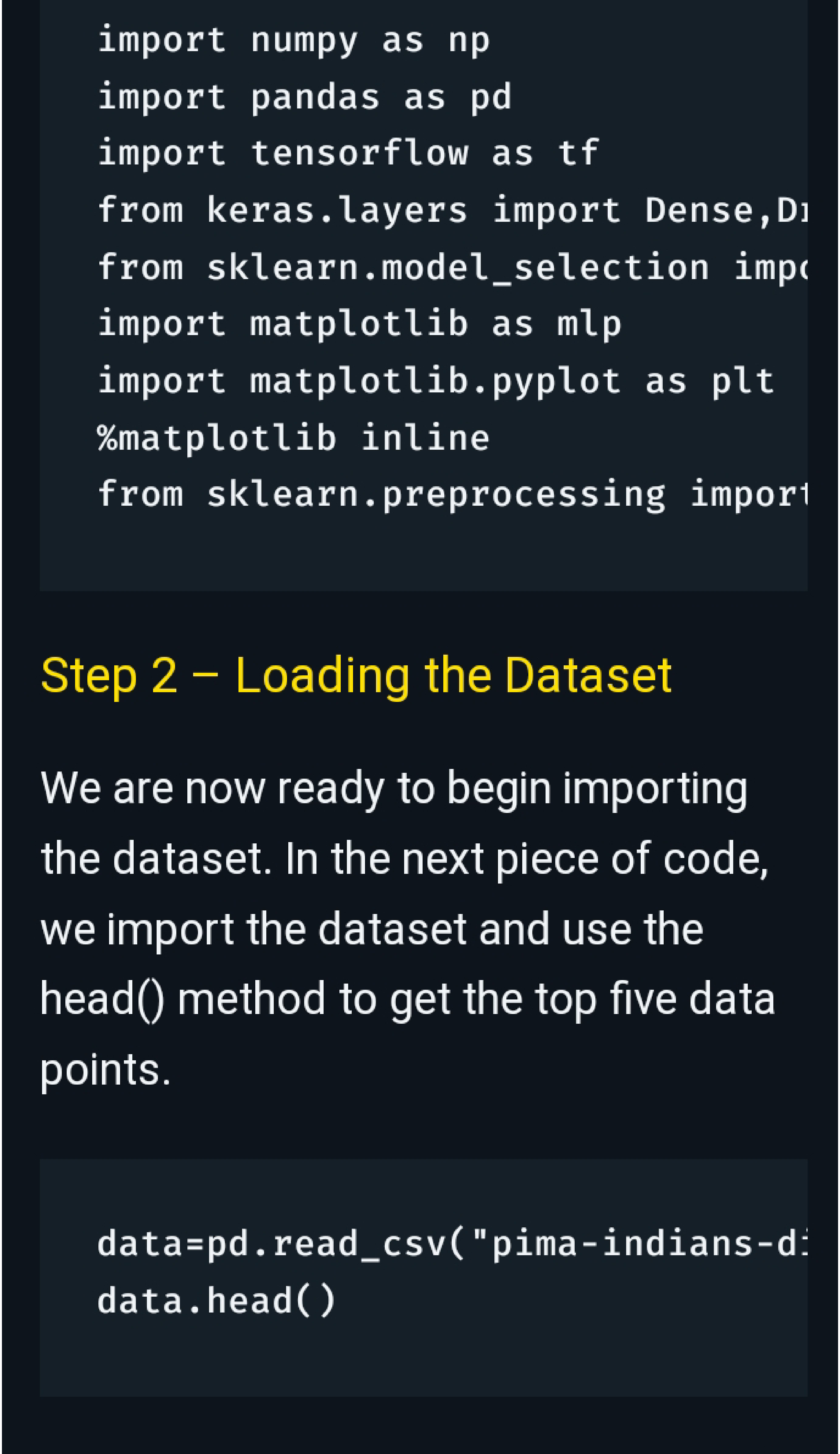
Step 1 – Importing Modules

Now, let’s import the necessary Python libraries into our notebook.

Keras API already includes Python’s TensorFlow deep learning package, which is critical in the diabetes prediction challenge.

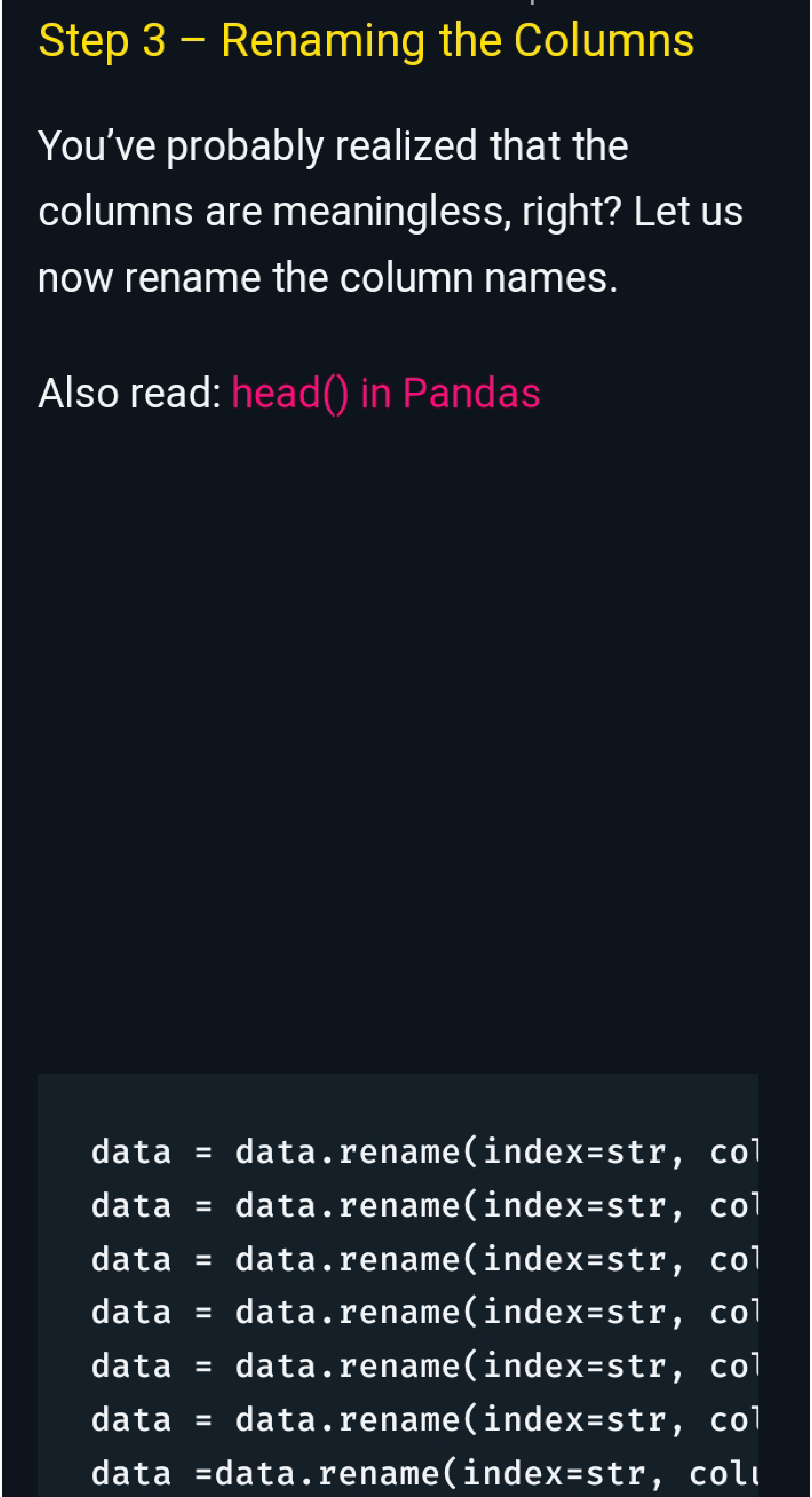


```
import numpy as np
import pandas as pd
import tensorflow as tf
from keras.layers import Dense,D
from sklearn.model_selection impo
import matplotlib as mlp
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn.preprocessing import
```

Step 2 – Loading the Dataset

We are now ready to begin importing the dataset. In the next piece of code, we import the dataset and use the head() method to get the top five data points.

```
data=pd.read_csv("pima-indians-di
data.head()
```



Step 3 – Renaming the Columns

You’ve probably realized that the columns are meaningless, right? Let us now rename the column names.

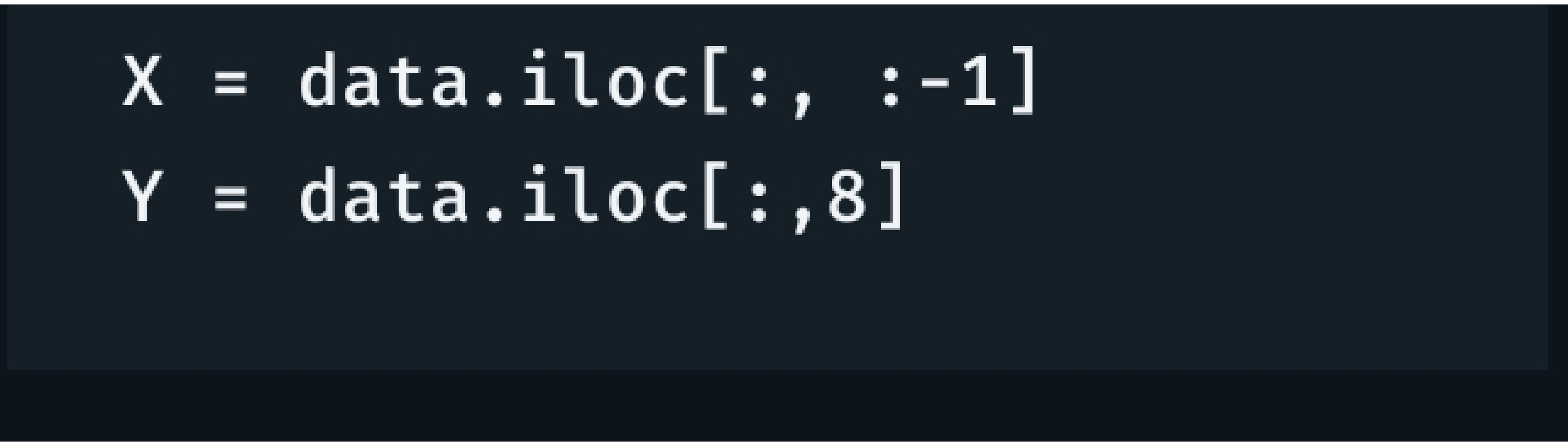
Also read: head() in Pandas

```
data = data.rename(index=str, col
data = data.rename(index=str, col
data = data.rename(index=str, col
data = data.rename(index=str, col
data = data.rename(index=str, col
data = data.rename(index=str, col
data =data.rename(index=str, colu
data = data.rename(index=str, col
data =data.rename(index=str, colu
data = data.rename(index=str, col
data = data.rename(index=str, col

data.head()
```

Renamed Columns Diabetes Dataset Top5

Step 4 – Separating Inputs and Outputs

```
X = data.iloc[:, :-1]
Y = data.iloc[:,8]
```



```
X = data.iloc[:, :-1]
Y = data.iloc[:,8]
```

Step 5 – Train-Test Split of the Data

The next step involves the training and testing split into data and then standardizing the data to make computations simpler later on.

```
X_train_full, X_test, y_train_ful
X_train, X_valid, y_train, y_vali
```

```
from sklearn.preprocessing import
scaler = StandardScaler()
```

```
X_train_full, X_test, y_train_ful
X_train, X_valid, y_train, y_vali
```

```
from sklearn.preprocessing import
scaler = StandardScaler()
X_train = scaler.fit_transform(X_
X_valid = scaler.transform(X_vali
X_test = scaler.transform(X_test)
```

Step 6 – Building the Model

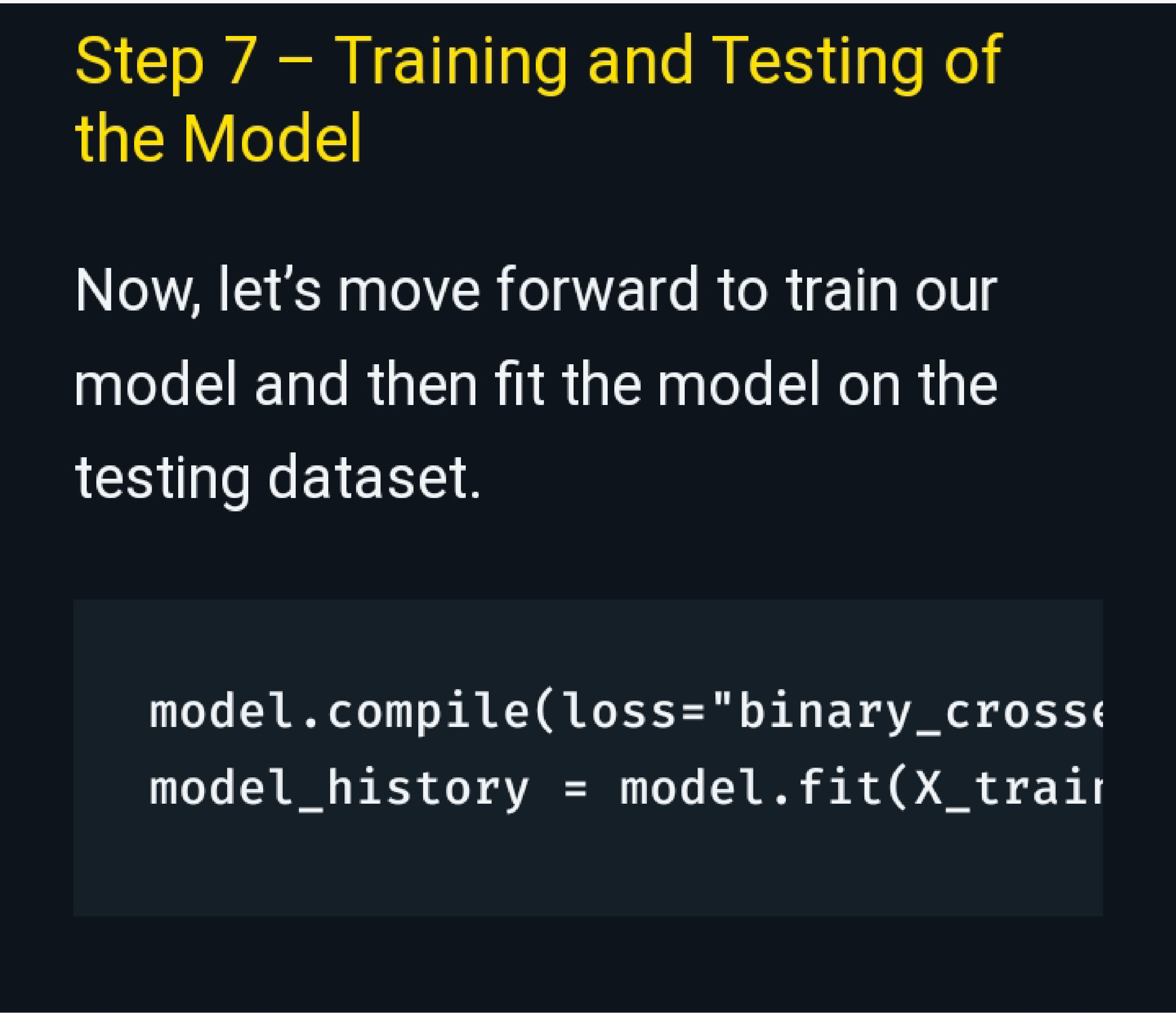
We start off by using a **random seed** to generate a pseudo-random number and setting it to the **tf graph**. Then, we will be using a sequential model, and also some dropout layers in the model to

```
np.random.seed(42)
tf.random.set_seed(42)

model=Sequential()
model.add(Dense(15,input_dim=8, a
model.add(Dense(10,activation='re
model.add(Dense(8,activation='rel
model.add(Dropout(0.25))
model.add(Dense(1, activation='si
```

Step 7 – Training and Testing of the Model

Now, let’s move forward to train our model and then fit the model on the testing dataset.

```
model.compile(loss="binary_crosse
model_history = model.fit(X_trai
```



Step 7 – Training and Testing of the Model

Now, let’s move forward to train our model and then fit the model on the testing dataset.

```
model.compile(loss="binary_crosse
model_history = model.fit(X_trai
```

Program Code for Model Training

```
      import numpy as np

      import pandas as pd from sklearn.model_selection

      import train_test_split from sklearn

      import svm from sklearn.metrics

      import accuracy_score import pickle

      diabetes_dataset = pd.read_csv('diabetes.csv')

      diabetes_dataset.head()

      diabetes_dataset.shape

      diabetes_dataset.describe()

      diabetes_dataset['Outcome'].value_counts()

      X = diabetes_dataset.drop(columns = 'Outcome', axis=1)

      Y = diabetes_dataset['Outcome']

      print(X)

      print(Y)

      X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size
=0.2, stratify=Y, random_state=2)

      print(X.shape, X_train.shape, X_test.shape)

      classifier = svm.SVC(kernel='linear')

      classifier.fit(X_train, Y_train)

      X_train_prediction = classifier.predict(X_train) training_data_accuracy
= accuracy_score(X_train_prediction, Y_train)

      print('Accuracy score of the training data : ', training_data_accuracy)
```

```
       X_test_prediction = classifier.predict(X_test) test_data_accuracy =
accuracy_score(X_test_prediction, Y_test)

       print('Accuracy score of the test data : ', test_data_accuracy)

       input_data = (5,166,72,19,175,25.8,0.587,51)

       input_data_as_numpy_array = np.asarray(input_data)
       input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)
prediction = classifier.predict(input_data_reshaped) print(prediction)

       if (prediction[0] == 0):

       print('The person is not diabetic') else: print('The person is
diabetic')

       filename = 'trained_model.sav'

       pickle.dump(classifier, open(filename, 'wb'))

       loaded_model = pickle.load(open('trained_model.sav', 'rb'))
```

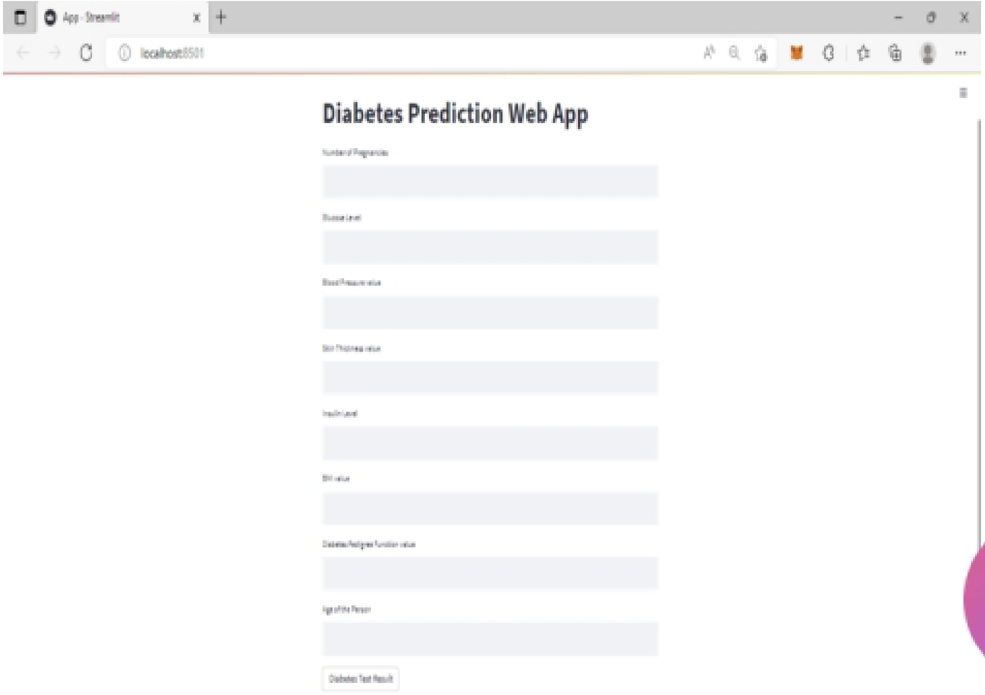
Program code for interacting with diabetes prediction

```
      import numpy as np
import pickle
import streamlit as st

# Load the saved model
loaded_model =
pickle.load(open('C:/Users/ELCOT/Downloads/trained_model.s
av', 'rb'))

# Create a function for Prediction
def diabetes_prediction(input_data):

  # Change the input_data to numpy array
```

```
  input_data_as_numpy_array = np.asarray(input_data)

  # Reshape the array as we are predicting for one instance
  input_data_reshaped =
input_data_as_numpy_array.reshape(1,-1)

  prediction = loaded_model.predict(input_data_reshaped)
  print(prediction)

  if (prediction[0] == 0):
   return 'The person is not diabetic'
  else:
   return 'The person is diabetic'

def main():
    input_data_as_numpy_array = np.asarray(input_data)

    # Reshape the array as we are predicting for one instance
    input_data_reshaped = input_data_as_numpy_array.reshape(1,-1)

    prediction = loaded_model.predict(input_data_reshaped)
    print(prediction)

    if (prediction[0] == 0):
      return 'The person is not diabetic'
    else:
      return 'The person is diabetic'

def main():

    # Give a title
    st.title('Diabetes Prediction Web App')

    # To get the input data from the user
    Pregnancies = st.text_input('Number of Pregnancies')
    Glucose = st.text_input('Glucose Level')
```

```
    BloodPressure = st.text_input('Blood Pressure value')
    SkinThickness = st.text_input('Skin Thickness value')
    Insulin = st.text_input('Insulin Level')
    BMI = st.text_input('BMI value')
    DiabetesPedigreeFunction = st.text_input('Diabetes Pedi
Function value')
    Age = st.text_input('Age of the Person')
    Age = st.text_input('Age of the Person')


    # Code for Prediction

    diagnosis = ''


    # Create a button for Prediction


    if st.button('Diabetes Test Result'):

        diagnosis = diabetes_prediction([Pregnancies, Glucose,
BloodPressure, SkinThickness, Insulin, BMI,
DiabetesPedigreeFunction, Age])


    st.success(diagnosis)


if __name__ == '__main__':

    main()
```

Output:



Diabetes Prediction Web App

Number of Pregnancies

Glucose Level

Blood Pressure value

Skin Thickness value

Insulin Level

BMI value

Diabetes Pedigree Function value

Age of the Person