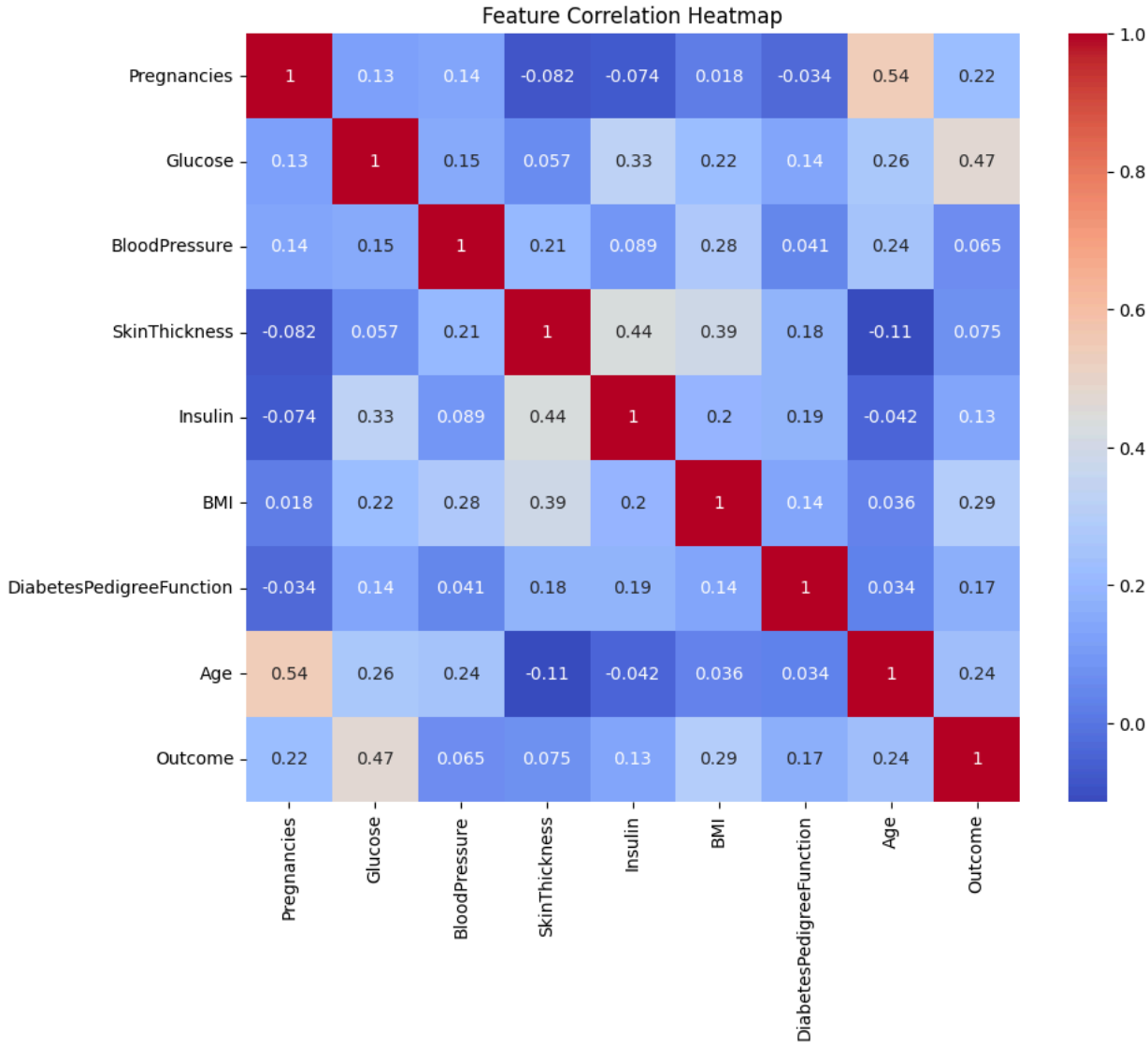
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense


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

print(df.head())
print("\nInfo of dataset:\n", df.info())
print("\nShape of dataset:", df.shape)
print("\nMissing values:\n", df.isnull().sum())
```

```
   Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin   BMI  \
0            6      148             72             35        0  33.6
1            1       85             66             29        0  26.6
2            8      183             64              0        0  23.3
3            1       89             66             23       94  28.1
4            0      137             40             35      168  43.1

   DiabetesPedigreeFunction  Age  Outcome
0                     0.627   50        1
1                     0.351   31        0
2                     0.672   32        1
3                     0.167   21        0
4                     2.288   33        1
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Pregnancies               768 non-null    int64
 1   Glucose                   768 non-null    int64
 2   BloodPressure             768 non-null    int64
 3   SkinThickness             768 non-null    int64
 4   Insulin                   768 non-null    int64
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB

Info of dataset:
 None

Shape of dataset: (768, 9)

Missing values:
 Pregnancies                 0
Glucose                     0
BloodPressure               0
SkinThickness               0
Insulin                     0
BMI                         0
DiabetesPedigreeFunction    0
Age                         0
Outcome                     0
dtype: int64
```

```
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title("Feature Correlation Heatmap")
plt.show()
```




```
X = df.drop('Outcome', axis=1)
y = df['Outcome']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
model = Sequential()

model.add(Dense(16, activation='relu', input_shape=(X_train.shape[1],)))

model.add(Dense(8, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

model.summary()
```

```
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/dense.py:93: UserWarning: Do not pass an `input_shape`/`input_dim` arg
  super().__init__(activity_regularizer=activity_regularizer, **kwargs)
Model: "sequential"
```

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 16)	144
dense_1 (Dense)	(None, 8)	136
dense_2 (Dense)	(None, 1)	9

```
Total params: 289 (1.13 KB)
Trainable params: 289 (1.13 KB)
Non-trainable params: 0 (0.00 B)
```

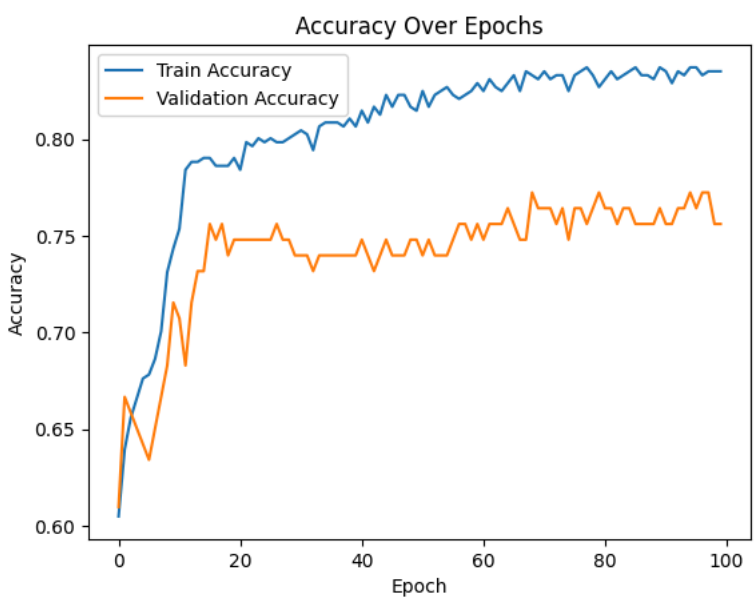
```
history = model.fit(X_train, y_train, epochs=100, batch_size=16, validation_split=0.2)
```

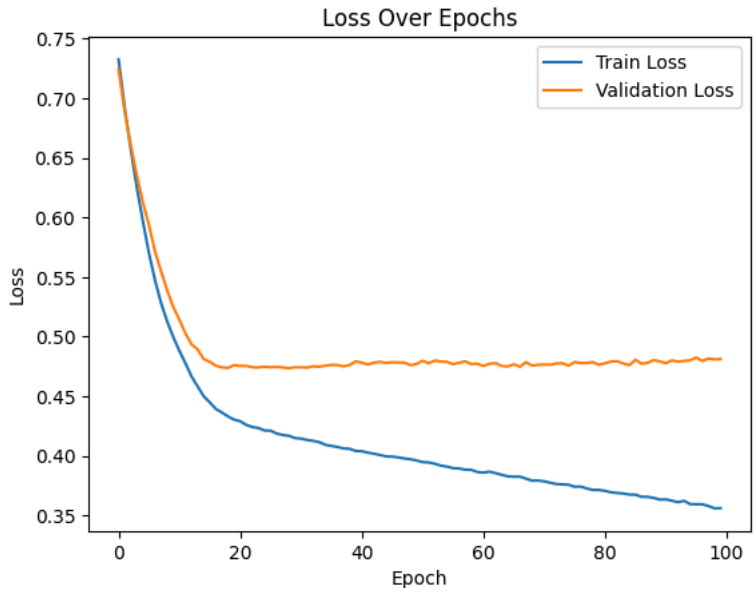
```
Epoch 72/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8239 - loss: 0.3749 - val_accuracy: 0.7642 - val_loss: 0.4765
Epoch 73/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8421 - loss: 0.3661 - val_accuracy: 0.7561 - val_loss: 0.4776
Epoch 74/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8270 - loss: 0.3944 - val_accuracy: 0.7642 - val_loss: 0.4777
Epoch 75/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8062 - loss: 0.3868 - val_accuracy: 0.7480 - val_loss: 0.4757
Epoch 76/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8284 - loss: 0.3634 - val_accuracy: 0.7642 - val_loss: 0.4788
Epoch 77/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8219 - loss: 0.3842 - val_accuracy: 0.7642 - val_loss: 0.4779
Epoch 78/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8328 - loss: 0.3751 - val_accuracy: 0.7561 - val_loss: 0.4781
Epoch 79/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8573 - loss: 0.3344 - val_accuracy: 0.7642 - val_loss: 0.4787
Epoch 80/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8427 - loss: 0.3458 - val_accuracy: 0.7724 - val_loss: 0.4764
Epoch 81/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8274 - loss: 0.3688 - val_accuracy: 0.7642 - val_loss: 0.4778
Epoch 82/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8194 - loss: 0.3558 - val_accuracy: 0.7642 - val_loss: 0.4792
Epoch 83/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8336 - loss: 0.3511 - val_accuracy: 0.7561 - val_loss: 0.4794
Epoch 84/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8540 - loss: 0.3314 - val_accuracy: 0.7642 - val_loss: 0.4775
Epoch 85/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8326 - loss: 0.3609 - val_accuracy: 0.7642 - val_loss: 0.4761
Epoch 86/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8174 - loss: 0.3974 - val_accuracy: 0.7561 - val_loss: 0.4806
Epoch 87/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.8503 - loss: 0.3374 - val_accuracy: 0.7561 - val_loss: 0.4774
Epoch 88/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8036 - loss: 0.4007 - val_accuracy: 0.7561 - val_loss: 0.4781
Epoch 89/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8409 - loss: 0.3442 - val_accuracy: 0.7561 - val_loss: 0.4803
Epoch 90/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8308 - loss: 0.3648 - val_accuracy: 0.7642 - val_loss: 0.4793
Epoch 91/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8504 - loss: 0.3488 - val_accuracy: 0.7561 - val_loss: 0.4778
Epoch 92/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8078 - loss: 0.3790 - val_accuracy: 0.7561 - val_loss: 0.4801
Epoch 93/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8141 - loss: 0.3822 - val_accuracy: 0.7642 - val_loss: 0.4791
Epoch 94/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8394 - loss: 0.3631 - val_accuracy: 0.7642 - val_loss: 0.4797
Epoch 95/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8394 - loss: 0.3509 - val_accuracy: 0.7724 - val_loss: 0.4803
Epoch 96/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8490 - loss: 0.3699 - val_accuracy: 0.7642 - val_loss: 0.4826
Epoch 97/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8063 - loss: 0.3922 - val_accuracy: 0.7724 - val_loss: 0.4797
Epoch 98/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8116 - loss: 0.3742 - val_accuracy: 0.7724 - val_loss: 0.4817
Epoch 99/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8117 - loss: 0.3977 - val_accuracy: 0.7561 - val_loss: 0.4810
Epoch 100/100
31/31 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.8067 - loss: 0.4036 - val_accuracy: 0.7561 - val_loss: 0.4812
```

```
loss, accuracy = model.evaluate(X_test, y_test)
print(f"\nTest Accuracy: {accuracy:.4f}")
```

```
5/5 ━━━━━━━━━━━━━━━━━━━━ 0s 7ms/step - accuracy: 0.7507 - loss: 0.5809

Test Accuracy: 0.7403
```

```
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.title("Accuracy Over Epochs")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()
plt.show()

plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.title("Loss Over Epochs")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend()
plt.show()
```






```
sample = np.array([[5, 116, 74, 0, 0, 25.6, 0.201, 30]])
sample = scaler.transform(sample)

prediction = model.predict(sample)
result = "Diabetic" if prediction[0][0] > 0.5 else "Non-diabetic"
print(f"\nPrediction: {result}")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 74ms/step
```