Introduction

This assignment is initiated because of the Week 3 ANN workshop at Fontys UAS.

Objective

Construct, train and test an artificial neural network using a dataset of your own choice. Try different settings for two or more hyperparameters and investigate the effect on learning. Write a Jupyter notebook which contains your python code and in which you describe your approach and results. In your notebook, you should describe your dataset and add a reference to the source of your dataset. Also, include references to any source code or tutorials that you used to write your code. If your neural network is aimed at classification, you should create a confusion matrix and discuss the results. Also reflect on the knowledge and skills you acquired on artificial neural networks.

Assignment Idea

The goal of this assignment is to test how to work with Artificial Neural Networks (ANN). The dataset chosen for this is taken from the Kaggle's website.

Assignment Goal

Diabetes is one of the major diseases of the population across the world. Diabetes is a chronic disease that occurs either when the pancreas does not produce enough insulin or when the body cannot efficiently use the insulin it produces. In 2014, 8.5% of adults aged 18 years and older had diabetes. In 2012, diabetes was the direct cause of 1.5 million deaths and high blood glucose was the cause of another 2.2 million deaths. Over the time, diabetes can damage the heart, blood vessels, eyes, kidneys, and nerves. Early diagnosis can be made through a relatively inexpensive method of computation. In this notebook the ANN model is used to analyze and make the diabetes prediction model.

The Dataset

In this dataset, all patients are females at least 21 years old of Pima Indian heritage. The total number of instances is 768, which is completely used in this study. It contains 8 attributes plus one class (Label) column. Each attribute is numeric-valued; attributes of this dataset are as follows:

1. Number of times pregnant
2. Plasma glucose concentration at 2 h in an oral glucose tolerance test
3. Diastolic blood pressure (mm Hg)
4. Triceps skinfold thickness (mm)
5. 2-hour serum insulin (mu U/ml)
6. Body mass index (weight in kg/(height in m2)
7. Diabetes pedigree function

8. Age (years)

9. Class variable (0 or 1).

(Class value 1 is interpreted as "tested positive for diabetes").

Import Libraries

In [1]:

```
import pandas as pd
import numpy as np
import tensorflow as tf
import seaborn as sns
import matplotlib.pyplot as plt
import warnings;
warnings.simplefilter('ignore')
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import Input, Dense
from tensorflow.keras.models import Sequential
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
%matplotlib inline
```

Import the data

In [2]:

```
df = pd.read_csv("diabetes.csv")
```

In [3]:

```
df.head(10)
```

Out[3]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

I can already notice that there are null values in the form of zero's.

In [4]:

```
df.isnull().sum()
```

Out[4]:

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

The values show that we do not have any missing data

In [5]:

```
df.dtypes
```

Out[5]:

```
Pregnancies                   int64
Glucose                       int64
BloodPressure                 int64
SkinThickness                 int64
Insulin                       int64
BMI                         float64
DiabetesPedigreeFunction    float64
Age                           int64
Outcome                       int64
dtype: object
```

In [6]:

```
df.info()
```

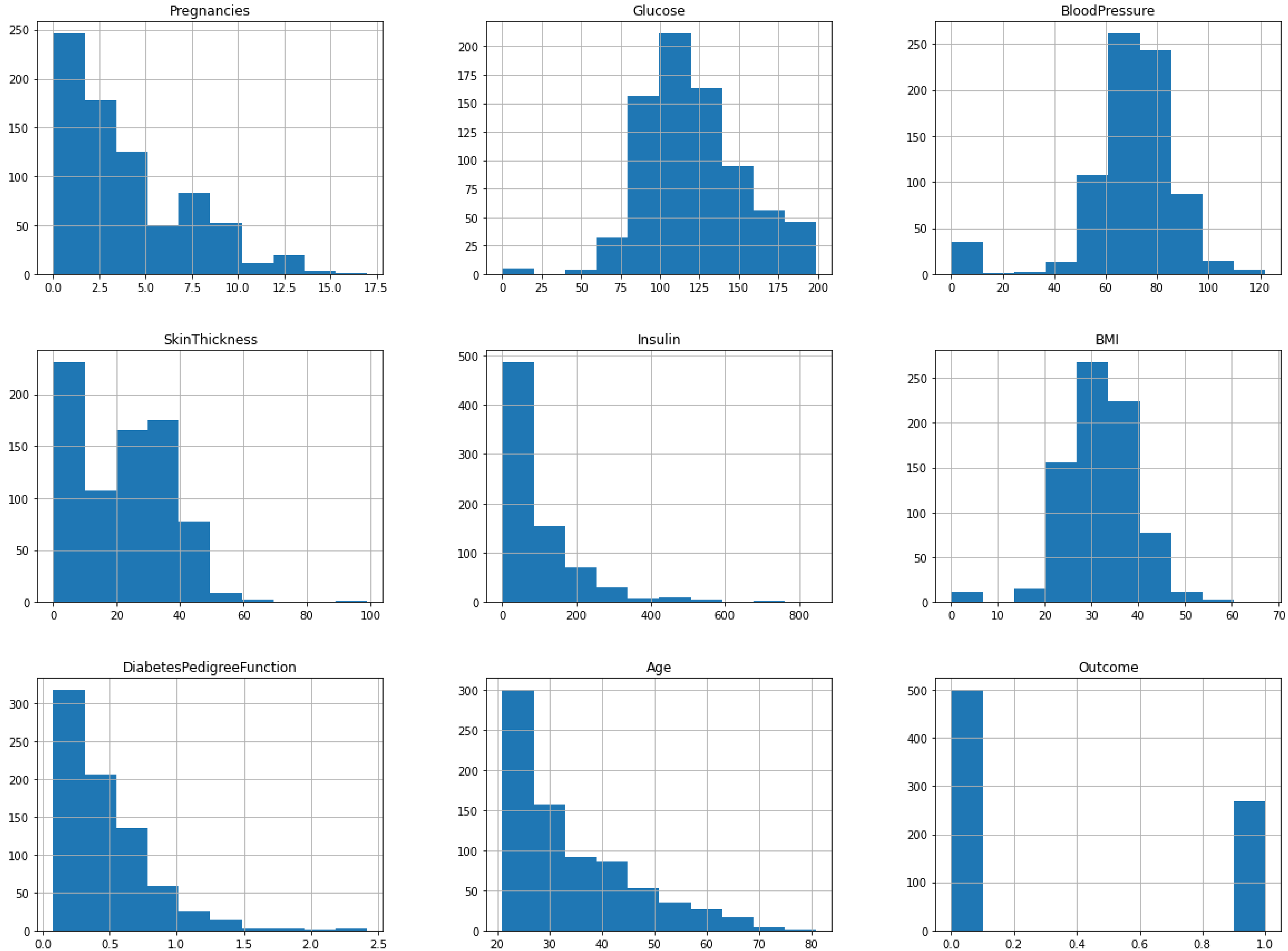
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   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
```

In [7]:

```
# I'll take a look at the distribution of the different parameters
df.hist(figsize=(20,15))
```

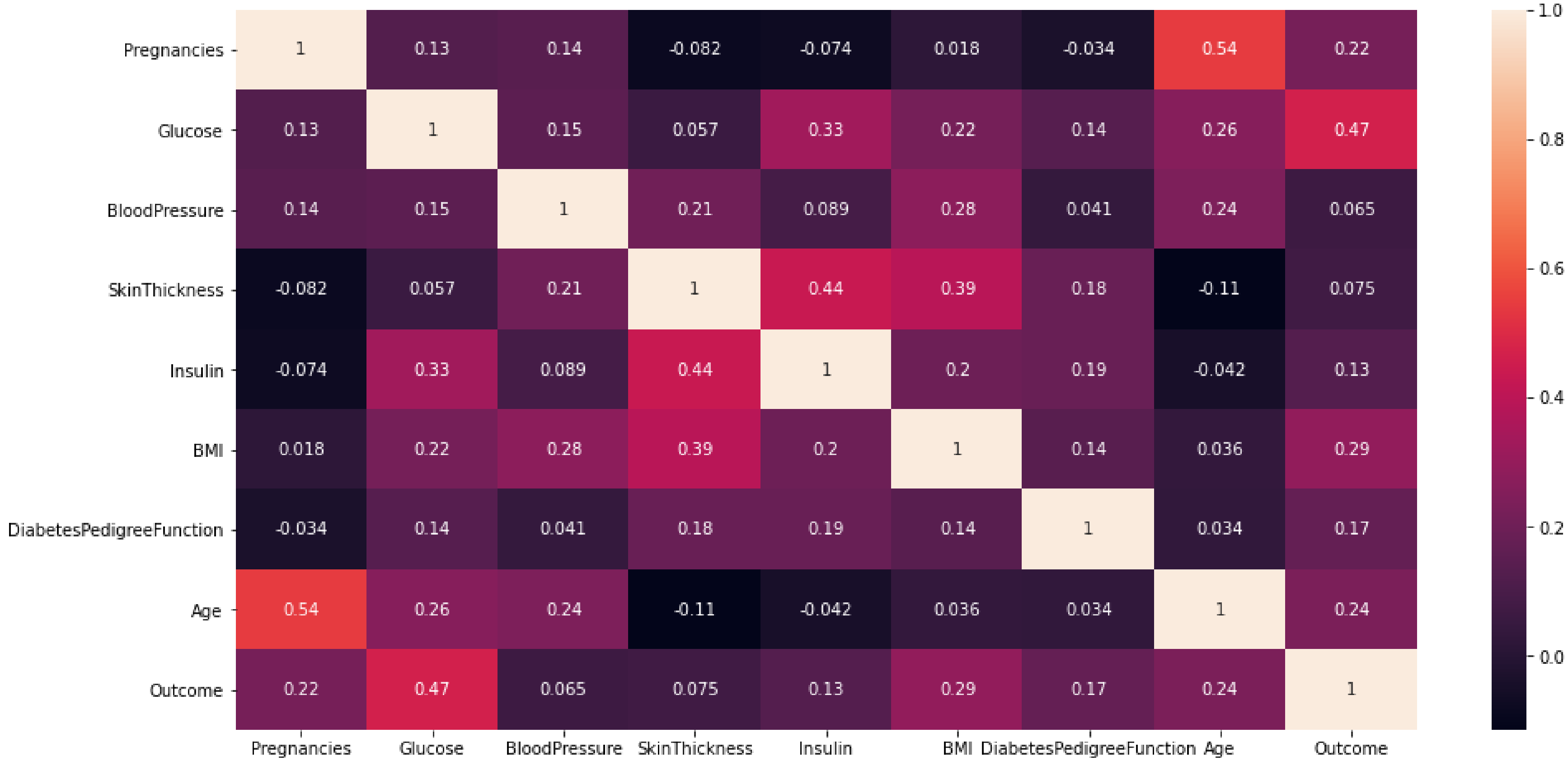
Out[7]:

```
array([[<AxesSubplot:title={'center':'Pregnancies'}>,
        <AxesSubplot:title={'center':'Glucose'}>,
        <AxesSubplot:title={'center':'BloodPressure'}>],
       [<AxesSubplot:title={'center':'SkinThickness'}>,
        <AxesSubplot:title={'center':'Insulin'}>,
        <AxesSubplot:title={'center':'BMI'}>],
       [<AxesSubplot:title={'center':'DiabetesPedigreeFunction'}>,
        <AxesSubplot:title={'center':'Age'}>,
        <AxesSubplot:title={'center':'Outcome'}>]], dtype=object)
```



In [8]:

```
plt.figure(1 , figsize = (16 , 8))
cor = sns.heatmap(df.corr(), annot = True)
```



Insights: There appears to be no strong correlations among the numeric features.

However, there are some slight correlations of Pregnancies, Glucose, BMI, Age with the outcome.

Check for missing values

In [9]:
```
#to determine presence of missing values in dataset
df[["Glucose", "SkinThickness","Insulin",
    "BloodPressure","BMI"]] =  df[["Glucose","SkinThickness", "Insulin","BloodPressure","BMI"]].replace(0, np.NaN)
```

In [10]:
```
df.isnull().sum()[1:6]
```

Out[10]:
```
Glucose           5
BloodPressure    35
```

```
SkinThickness    227
Insulin          374
BMI               11
dtype: int64
```

- Important Observations:

    - As I mentioned above, tt seems like null values are present in the form of zero's.
    - It's impossible to have Glucose, Blood Pressure, SkinThickness, Insulin and BMI to be zero.

In [11]:

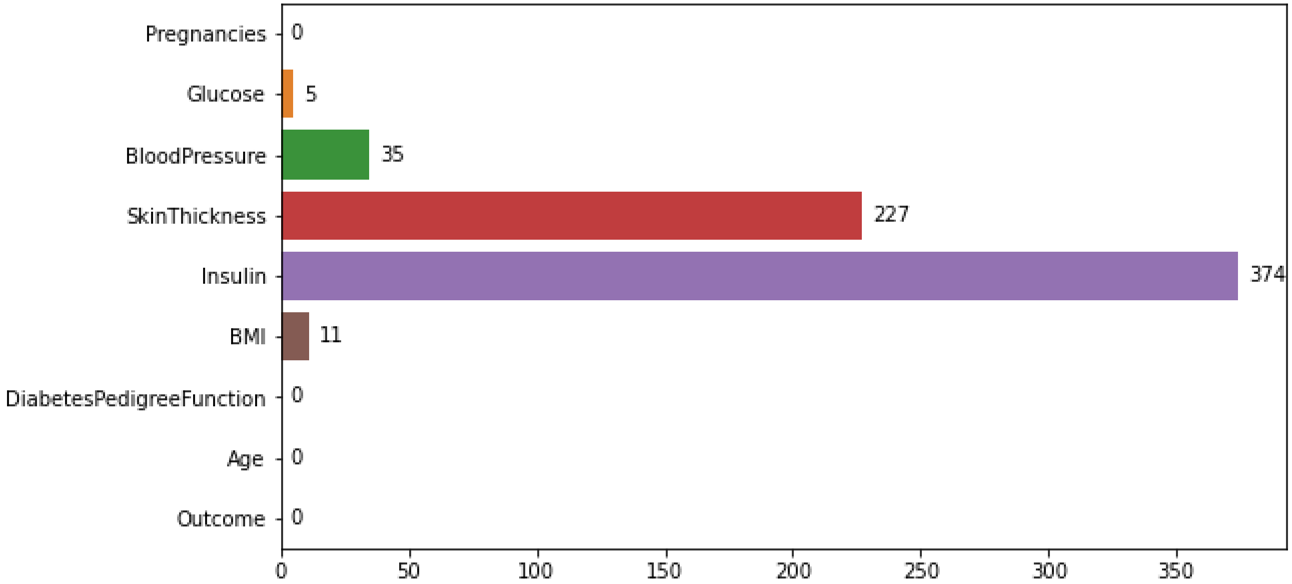
```
df.describe().T
```

Out[11]:

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.0000	6.00000	17.00
Glucose	763.0	121.686763	30.535641	44.000	99.00000	117.0000	141.00000	199.00
BloodPressure	733.0	72.405184	12.382158	24.000	64.00000	72.0000	80.00000	122.00
SkinThickness	541.0	29.153420	10.476982	7.000	22.00000	29.0000	36.00000	99.00
Insulin	394.0	155.548223	118.775855	14.000	76.25000	125.0000	190.00000	846.00
BMI	757.0	32.457464	6.924988	18.200	27.50000	32.3000	36.60000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.3725	0.62625	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.0000	41.00000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.0000	1.00000	1.00

In [12]:

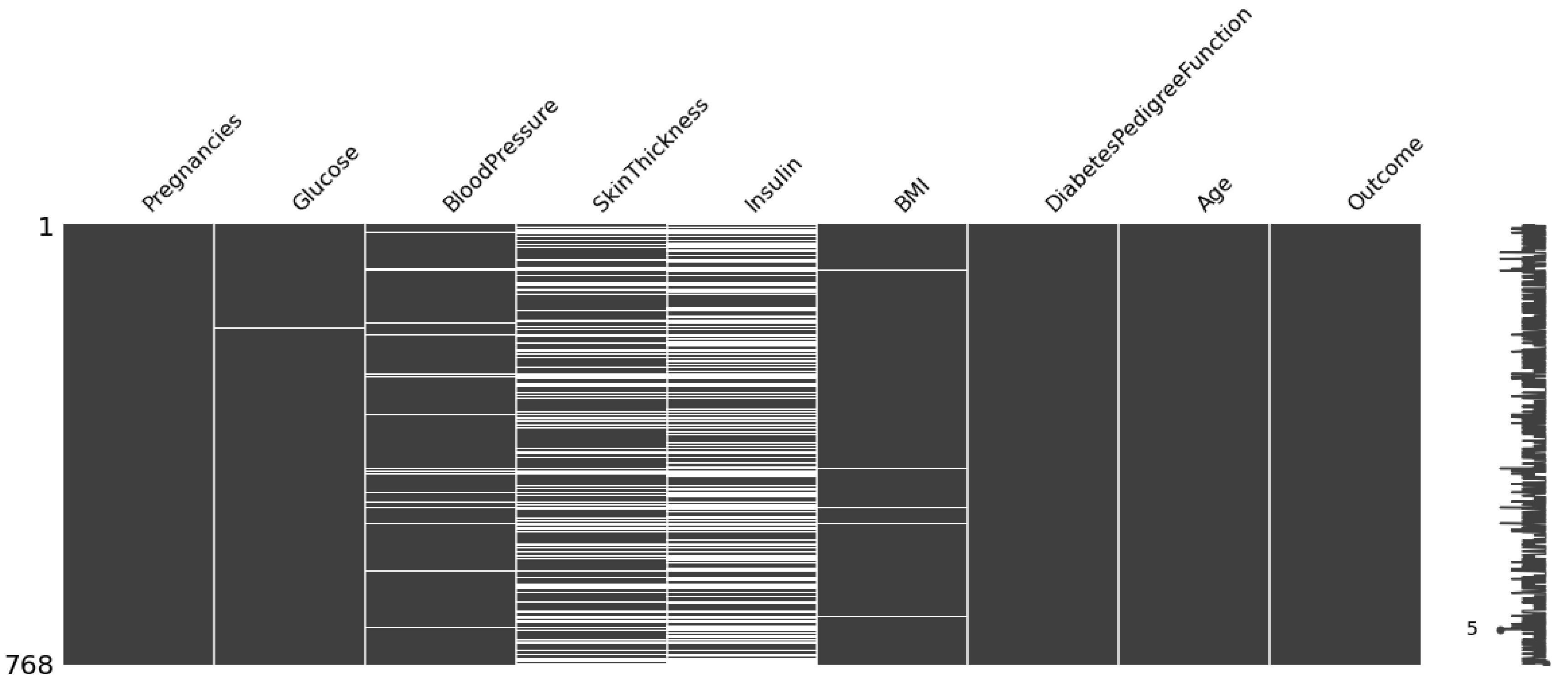
```
#Visualizing Null values
plt.figure(figsize=(9,5))
ax = sns.barplot(x=df.isna().sum(),
           y=df.columns, orient='h')
for p in ax.patches:
    ax.annotate(text=f"{p.get_width():.0f}",
                xy=(p.get_width(), p.get_y()+p.get_height()/2),
                xytext=(5, 0), textcoords='offset points',
                ha="left", va="center",
               )
plt.grid(False)
plt.show()
```



In [13]:

```
import missingno as mno
mno.matrix(df, figsize = (20, 6))
```

Out[13]: <AxesSubplot:>



From the above plot, we can notice that there are missing values in the Glucose, BloodPressure, SkinThickness, Insulin and BMI.

We can see there are lot of null values in SkinThickness and Insulin column. So, after Imputation the mean will change drastically.

In [14]:

```
#imputing mean instead of null values
for col in df:
    df[col].replace(np.nan, df[col].mean(), inplace=True)
```

In [15]:

```
df.describe().T
```

Out[15]:

	count	mean	std	min	25%	50%	75%	max
Pregnancies	768.0	3.845052	3.369578	0.000	1.00000	3.000000	6.000000	17.00
Glucose	768.0	121.686763	30.435949	44.000	99.75000	117.000000	140.250000	199.00
BloodPressure	768.0	72.405184	12.096346	24.000	64.00000	72.202592	80.000000	122.00
SkinThickness	768.0	29.153420	8.790942	7.000	25.00000	29.153420	32.000000	99.00

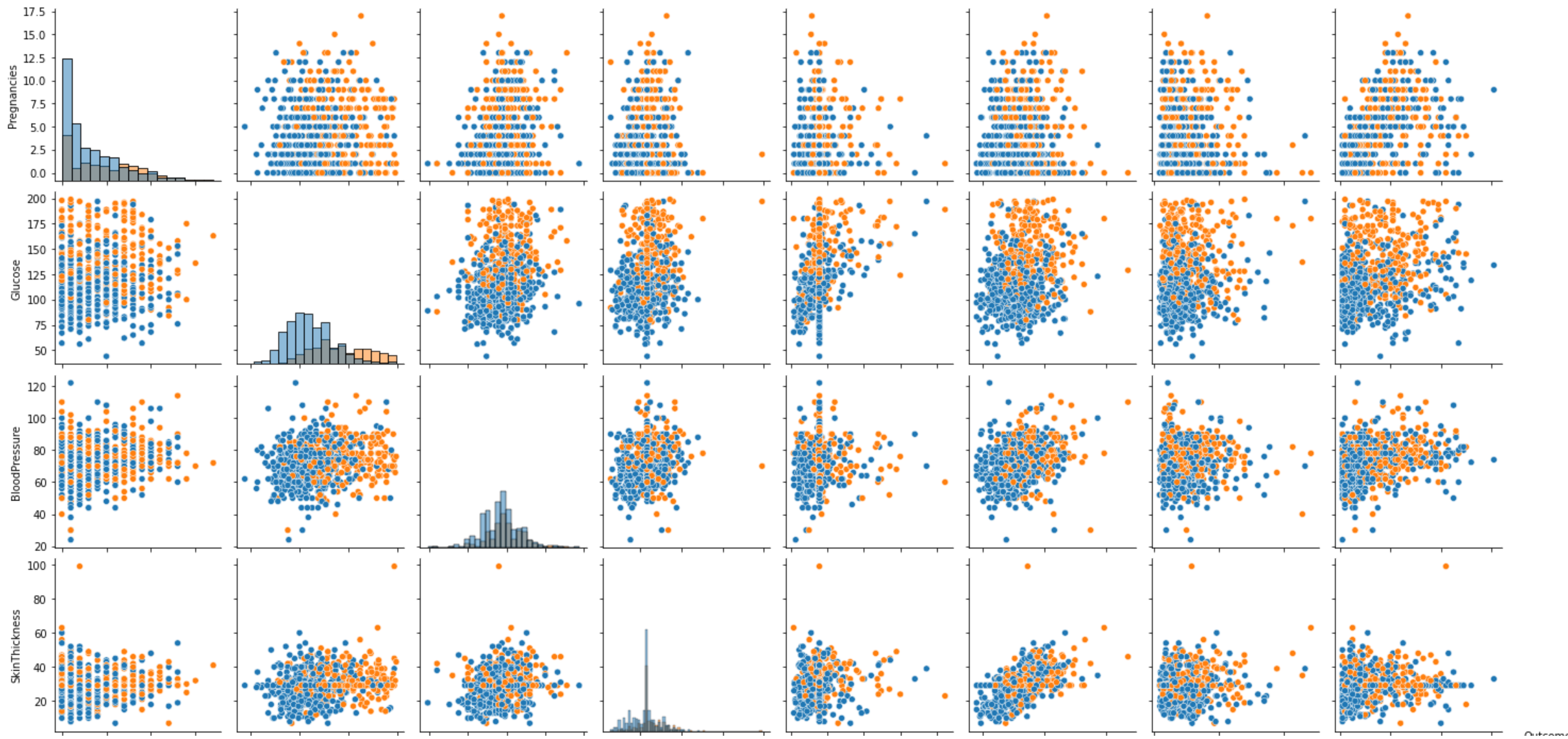
	count	mean	std	min	25%	50%	75%	max
Insulin	768.0	155.548223	85.021108	14.000	121.50000	155.548223	155.548223	846.00
BMI	768.0	32.457464	6.875151	18.200	27.50000	32.400000	36.600000	67.10
DiabetesPedigreeFunction	768.0	0.471876	0.331329	0.078	0.24375	0.372500	0.626250	2.42
Age	768.0	33.240885	11.760232	21.000	24.00000	29.000000	41.000000	81.00
Outcome	768.0	0.348958	0.476951	0.000	0.00000	0.000000	1.000000	1.00

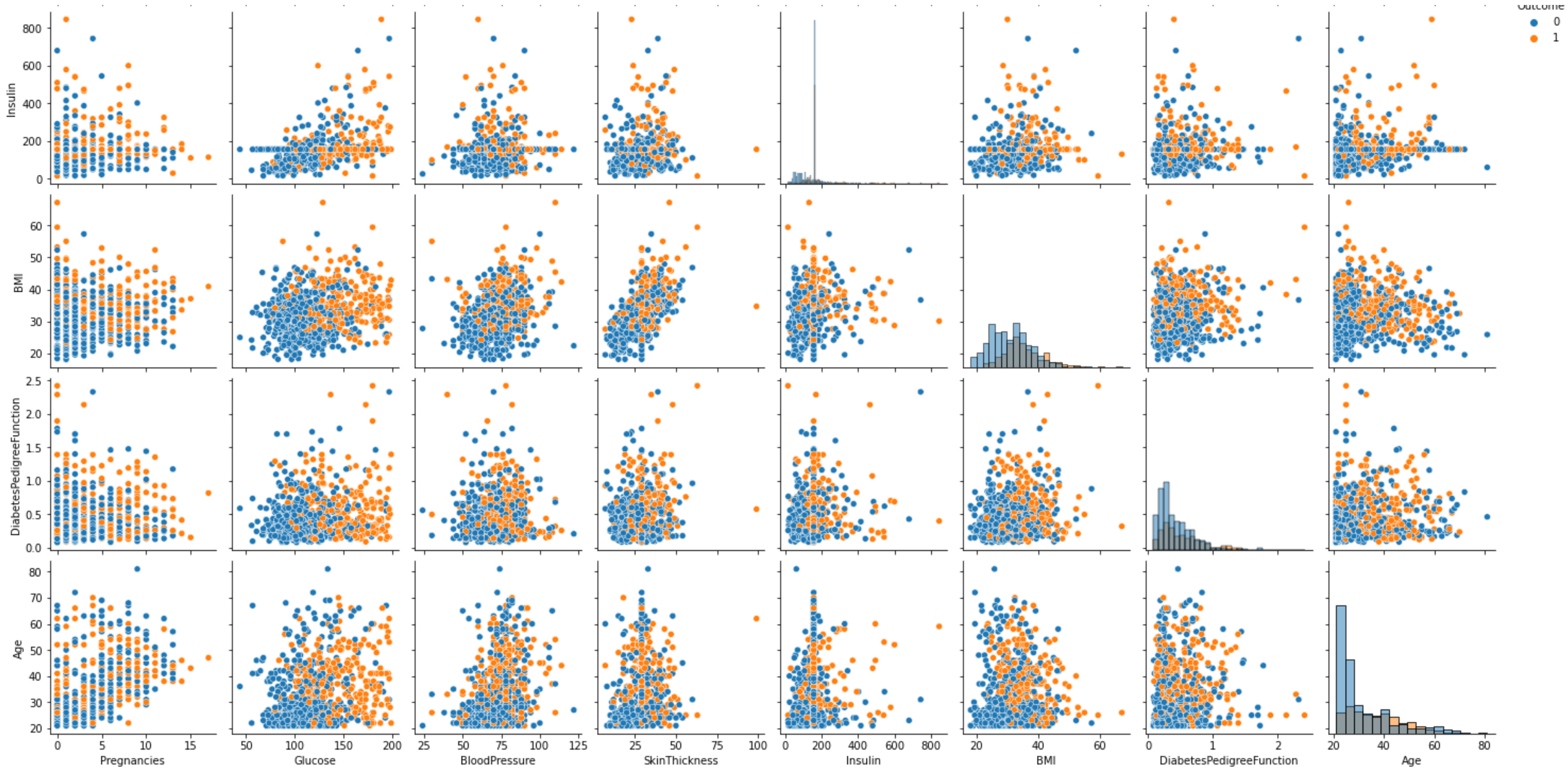
I can now notice that the minimum value for insulin has increased.

In [16]:

```
#Plot pairwise relationships in a dataset
plt.figure(figsize=(20,20))
sns.pairplot(data=df, hue="Outcome", diag_kind="hist")
plt.show()
```

```
<Figure size 1440x1440 with 0 Axes>
```





In [17]:
```python
#distribution of outcomes
df["Outcome"].value_counts()
```

Out[17]:
```
0    500
1    268
Name: Outcome, dtype: int64
```

In [18]:
```python
# Replacing missing values
num_col = ['Glucose', 'BloodPressure',  'SkinThickness', 'Insulin', 'BMI']
for col in num_col:
    df[col]=pd.to_numeric(df[col])
    df[col].fillna(df[col].mean(), inplace=True)
df.head()
```

Out[18]:

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148.0	72.0	35.00000	155.548223	33.6	0.627	50	1
1	1	85.0	66.0	29.00000	155.548223	26.6	0.351	31	0
2	8	183.0	64.0	29.15342	155.548223	23.3	0.672	32	1
3	1	89.0	66.0	23.00000	94.000000	28.1	0.167	21	0
4	0	137.0	40.0	35.00000	168.000000	43.1	2.288	33	1

Modeling

Splitting the dataset

In [19]:

```
X=df.drop(["Outcome"],axis='columns') # dropping the target variable
y=df["Outcome"] # keeping the target variable
```

In [20]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=42)
```

Standardize the data

In [21]:

```
# Import `StandardScaler` from `sklearn.preprocessing`
from sklearn.preprocessing import StandardScaler

# Define the scaler
scaler = StandardScaler().fit(X_train)

# Scale the train set
X_train = scaler.transform(X_train)

# Scale the test set
X_test = scaler.transform(X_test)
```

In [22]:

```
from sklearn.utils import shuffle
```

```
X_train, y_train = shuffle(X_train,y_train, random_state=14)
```

In [23]:

```
print("Train data length:",len(X_train))
print("Test data length:",len(X_test))
X_train.shape[1]
```

```
Train data length: 614
Test data length: 154
```

Out[23]: 8

In [24]:

```
# Initializing the model
model = keras.Sequential([
    keras.layers.Dense(64, input_shape=(8,), activation='relu'),
    keras.layers.Dense(32, activation='relu'),
    keras.layers.Dense(16, activation='relu'),
    keras.layers.Dense(1, activation='sigmoid')
])
```

In [25]:

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

In [26]:

```
# Fitting the model to the Training set
hist = model.fit(X_train, y_train, epochs=100, validation_split=0.2)
```

```
Epoch 1/100
16/16 [==============================] - 0s 13ms/step - loss: 0.6051 - accuracy: 0.7128 - val_loss: 0.5768 - val_accurac
y: 0.7236
Epoch 2/100
16/16 [==============================] - 0s 2ms/step - loss: 0.5322 - accuracy: 0.7536 - val_loss: 0.5393 - val_accuracy:
0.7236
Epoch 3/100
16/16 [==============================] - 0s 2ms/step - loss: 0.4871 - accuracy: 0.7699 - val_loss: 0.5159 - val_accuracy:
0.7154
Epoch 4/100
16/16 [==============================] - 0s 2ms/step - loss: 0.4608 - accuracy: 0.7841 - val_loss: 0.5035 - val_accuracy:
0.7236
Epoch 5/100
16/16 [==============================] - 0s 3ms/step - loss: 0.4417 - accuracy: 0.7943 - val_loss: 0.4947 - val_accuracy:
0.7317
Epoch 6/100
16/16 [==============================] - 0s 3ms/step - loss: 0.4316 - accuracy: 0.7862 - val_loss: 0.4884 - val_accuracy:
```

```
0.7317
Epoch 7/100
16/16 [==============================] - 0s 3ms/step - loss: 0.4191 - accuracy: 0.7943 - val_loss: 0.4908 - val_accuracy:
0.7480
Epoch 8/100
16/16 [==============================] - 0s 3ms/step - loss: 0.4121 - accuracy: 0.7943 - val_loss: 0.4836 - val_accuracy:
0.7398
Epoch 9/100
16/16 [==============================] - 0s 3ms/step - loss: 0.4050 - accuracy: 0.7963 - val_loss: 0.4802 - val_accuracy:
0.7561
Epoch 10/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3995 - accuracy: 0.7963 - val_loss: 0.4788 - val_accuracy:
0.7642
Epoch 11/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3945 - accuracy: 0.8024 - val_loss: 0.4773 - val_accuracy:
0.7561
Epoch 12/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3887 - accuracy: 0.8004 - val_loss: 0.4800 - val_accuracy:
0.7561
Epoch 13/100
16/16 [==============================] - ETA: 0s - loss: 0.4869 - accuracy: 0.71 - 0s 3ms/step - loss: 0.3834 - accuracy:
0.8167 - val_loss: 0.4869 - val_accuracy: 0.7561
Epoch 14/100
16/16 [==============================] - 0s 2ms/step - loss: 0.3772 - accuracy: 0.8126 - val_loss: 0.4821 - val_accuracy:
0.7561
Epoch 15/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3740 - accuracy: 0.8147 - val_loss: 0.4818 - val_accuracy:
0.7724
Epoch 16/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3688 - accuracy: 0.8167 - val_loss: 0.4839 - val_accuracy:
0.7724
Epoch 17/100
16/16 [==============================] - 0s 2ms/step - loss: 0.3661 - accuracy: 0.8208 - val_loss: 0.4866 - val_accuracy:
0.7724
Epoch 18/100
16/16 [==============================] - 0s 2ms/step - loss: 0.3618 - accuracy: 0.8147 - val_loss: 0.4901 - val_accuracy:
0.7561
Epoch 19/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3610 - accuracy: 0.8187 - val_loss: 0.4896 - val_accuracy:
0.7642
Epoch 20/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3575 - accuracy: 0.8248 - val_loss: 0.4928 - val_accuracy:
0.7642
Epoch 21/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3483 - accuracy: 0.8269 - val_loss: 0.4969 - val_accuracy:
0.7561
Epoch 22/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3457 - accuracy: 0.8187 - val_loss: 0.4965 - val_accuracy:
0.7805
```

```
Epoch 23/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3384 - accuracy: 0.8269 - val_loss: 0.5035 - val_accuracy:
0.7805
Epoch 24/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3373 - accuracy: 0.8330 - val_loss: 0.5084 - val_accuracy:
0.7724
Epoch 25/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3309 - accuracy: 0.8391 - val_loss: 0.5084 - val_accuracy:
0.7642
Epoch 26/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3295 - accuracy: 0.8350 - val_loss: 0.5105 - val_accuracy:
0.7561
Epoch 27/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3285 - accuracy: 0.8432 - val_loss: 0.5209 - val_accuracy:
0.7724
Epoch 28/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3199 - accuracy: 0.8391 - val_loss: 0.5214 - val_accuracy:
0.7724
Epoch 29/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3165 - accuracy: 0.8513 - val_loss: 0.5213 - val_accuracy:
0.7642
Epoch 30/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3093 - accuracy: 0.8493 - val_loss: 0.5347 - val_accuracy:
0.7724
Epoch 31/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3049 - accuracy: 0.8574 - val_loss: 0.5376 - val_accuracy:
0.7724
Epoch 32/100
16/16 [==============================] - 0s 3ms/step - loss: 0.3012 - accuracy: 0.8635 - val_loss: 0.5399 - val_accuracy:
0.7724
Epoch 33/100
16/16 [==============================] - 0s 3ms/step - loss: 0.2973 - accuracy: 0.8574 - val_loss: 0.5541 - val_accuracy:
0.7642
Epoch 34/100
16/16 [==============================] - 0s 3ms/step - loss: 0.2939 - accuracy: 0.8595 - val_loss: 0.5547 - val_accuracy:
0.7805
Epoch 35/100
16/16 [==============================] - 0s 3ms/step - loss: 0.2887 - accuracy: 0.8534 - val_loss: 0.5672 - val_accuracy:
0.7642
Epoch 36/100
16/16 [==============================] - 0s 3ms/step - loss: 0.2844 - accuracy: 0.8737 - val_loss: 0.5623 - val_accuracy:
0.7724
Epoch 37/100
16/16 [==============================] - 0s 3ms/step - loss: 0.2760 - accuracy: 0.8778 - val_loss: 0.5841 - val_accuracy:
0.7642
Epoch 38/100
16/16 [==============================] - 0s 3ms/step - loss: 0.2725 - accuracy: 0.8717 - val_loss: 0.5913 - val_accuracy:
0.7642
Epoch 39/100
```

```
16/16 [==============================] - 0s 3ms/step - loss: 0.2681 - accuracy: 0.8839 - val_loss: 0.5911 - val_accuracy:
0.7561
Epoch 40/100
16/16 [==============================] - 0s 3ms/step - loss: 0.2588 - accuracy: 0.8880 - val_loss: 0.6131 - val_accuracy:
0.7561
Epoch 41/100
16/16 [==============================] - 0s 3ms/step - loss: 0.2529 - accuracy: 0.8778 - val_loss: 0.6162 - val_accuracy:
0.7724
Epoch 42/100
16/16 [==============================] - 0s 3ms/step - loss: 0.2514 - accuracy: 0.8819 - val_loss: 0.6206 - val_accuracy:
0.7724
Epoch 43/100
16/16 [==============================] - 0s 3ms/step - loss: 0.2439 - accuracy: 0.8859 - val_loss: 0.6354 - val_accuracy:
0.7561
Epoch 44/100
16/16 [==============================] - 0s 3ms/step - loss: 0.2375 - accuracy: 0.8900 - val_loss: 0.6415 - val_accuracy:
0.7642
Epoch 45/100
16/16 [==============================] - 0s 3ms/step - loss: 0.2319 - accuracy: 0.8982 - val_loss: 0.6614 - val_accuracy:
0.7480
Epoch 46/100
16/16 [==============================] - 0s 3ms/step - loss: 0.2259 - accuracy: 0.8982 - val_loss: 0.6742 - val_accuracy:
0.7561
Epoch 47/100
16/16 [==============================] - 0s 3ms/step - loss: 0.2252 - accuracy: 0.9063 - val_loss: 0.6739 - val_accuracy:
0.7805
Epoch 48/100
16/16 [==============================] - 0s 3ms/step - loss: 0.2138 - accuracy: 0.9043 - val_loss: 0.6725 - val_accuracy:
0.7805
Epoch 49/100
16/16 [==============================] - 0s 3ms/step - loss: 0.2093 - accuracy: 0.9084 - val_loss: 0.7224 - val_accuracy:
0.7317
Epoch 50/100
16/16 [==============================] - 0s 2ms/step - loss: 0.2112 - accuracy: 0.8961 - val_loss: 0.7093 - val_accuracy:
0.7642
Epoch 51/100
16/16 [==============================] - 0s 2ms/step - loss: 0.2039 - accuracy: 0.9104 - val_loss: 0.7536 - val_accuracy:
0.7398
Epoch 52/100
16/16 [==============================] - 0s 2ms/step - loss: 0.1913 - accuracy: 0.9185 - val_loss: 0.7350 - val_accuracy:
0.7480
Epoch 53/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1868 - accuracy: 0.9226 - val_loss: 0.7584 - val_accuracy:
0.7561
Epoch 54/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1828 - accuracy: 0.9308 - val_loss: 0.7649 - val_accuracy:
0.7724
Epoch 55/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1804 - accuracy: 0.9226 - val_loss: 0.7957 - val_accuracy:
```

```
0.7398
Epoch 56/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1722 - accuracy: 0.9287 - val_loss: 0.7810 - val_accuracy:
0.7642
Epoch 57/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1660 - accuracy: 0.9348 - val_loss: 0.8168 - val_accuracy:
0.7398
Epoch 58/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1640 - accuracy: 0.9246 - val_loss: 0.8354 - val_accuracy:
0.7480
Epoch 59/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1561 - accuracy: 0.9511 - val_loss: 0.8295 - val_accuracy:
0.7480
Epoch 60/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1537 - accuracy: 0.9430 - val_loss: 0.8699 - val_accuracy:
0.7642
Epoch 61/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1488 - accuracy: 0.9470 - val_loss: 0.8836 - val_accuracy:
0.7480
Epoch 62/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1434 - accuracy: 0.9450 - val_loss: 0.8734 - val_accuracy:
0.7480
Epoch 63/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1381 - accuracy: 0.9450 - val_loss: 0.9113 - val_accuracy:
0.7642
Epoch 64/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1302 - accuracy: 0.9532 - val_loss: 0.9106 - val_accuracy:
0.7398
Epoch 65/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1294 - accuracy: 0.9593 - val_loss: 0.9415 - val_accuracy:
0.7561
Epoch 66/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1245 - accuracy: 0.9593 - val_loss: 0.9436 - val_accuracy:
0.7236
Epoch 67/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1209 - accuracy: 0.9654 - val_loss: 0.9984 - val_accuracy:
0.7154
Epoch 68/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1178 - accuracy: 0.9633 - val_loss: 0.9830 - val_accuracy:
0.7480
Epoch 69/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1154 - accuracy: 0.9674 - val_loss: 1.0217 - val_accuracy:
0.7236
Epoch 70/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1100 - accuracy: 0.9654 - val_loss: 1.0255 - val_accuracy:
0.7398
Epoch 71/100
16/16 [==============================] - 0s 3ms/step - loss: 0.1060 - accuracy: 0.9674 - val_loss: 1.0263 - val_accuracy:
0.7398
```

```
Epoch 72/100
16/16 [==============================] - 0s 2ms/step - loss: 0.1017 - accuracy: 0.9756 - val_loss: 1.0644 - val_accuracy:
0.7317
Epoch 73/100
16/16 [==============================] - 0s 2ms/step - loss: 0.0990 - accuracy: 0.9796 - val_loss: 1.0849 - val_accuracy:
0.7480
Epoch 74/100
16/16 [==============================] - 0s 2ms/step - loss: 0.0945 - accuracy: 0.9796 - val_loss: 1.1072 - val_accuracy:
0.7317
Epoch 75/100
16/16 [==============================] - 0s 2ms/step - loss: 0.0927 - accuracy: 0.9756 - val_loss: 1.1202 - val_accuracy:
0.7154
Epoch 76/100
16/16 [==============================] - 0s 2ms/step - loss: 0.0902 - accuracy: 0.9878 - val_loss: 1.1261 - val_accuracy:
0.7317
Epoch 77/100
16/16 [==============================] - 0s 3ms/step - loss: 0.0897 - accuracy: 0.9776 - val_loss: 1.0959 - val_accuracy:
0.7561
Epoch 78/100
16/16 [==============================] - 0s 2ms/step - loss: 0.0859 - accuracy: 0.9857 - val_loss: 1.1870 - val_accuracy:
0.7236
Epoch 79/100
16/16 [==============================] - 0s 2ms/step - loss: 0.0807 - accuracy: 0.9857 - val_loss: 1.2393 - val_accuracy:
0.7317
Epoch 80/100
16/16 [==============================] - 0s 2ms/step - loss: 0.0803 - accuracy: 0.9776 - val_loss: 1.1870 - val_accuracy:
0.7561
Epoch 81/100
16/16 [==============================] - 0s 3ms/step - loss: 0.0742 - accuracy: 0.9898 - val_loss: 1.2290 - val_accuracy:
0.7154
Epoch 82/100
16/16 [==============================] - 0s 2ms/step - loss: 0.0785 - accuracy: 0.9817 - val_loss: 1.3131 - val_accuracy:
0.7480
Epoch 83/100
16/16 [==============================] - 0s 2ms/step - loss: 0.0730 - accuracy: 0.9817 - val_loss: 1.2616 - val_accuracy:
0.7317
Epoch 84/100
16/16 [==============================] - 0s 2ms/step - loss: 0.0680 - accuracy: 0.9898 - val_loss: 1.2619 - val_accuracy:
0.7561
Epoch 85/100
16/16 [==============================] - 0s 2ms/step - loss: 0.0655 - accuracy: 0.9898 - val_loss: 1.2966 - val_accuracy:
0.7154
Epoch 86/100
16/16 [==============================] - 0s 2ms/step - loss: 0.0628 - accuracy: 0.9919 - val_loss: 1.3034 - val_accuracy:
0.7561
Epoch 87/100
16/16 [==============================] - 0s 3ms/step - loss: 0.0598 - accuracy: 0.9919 - val_loss: 1.3444 - val_accuracy:
0.7236
Epoch 88/100
```

```
16/16 [==============================] - 0s 2ms/step - loss: 0.0590 - accuracy: 0.9919 - val_loss: 1.3381 - val_accuracy:
0.7398
Epoch 89/100
16/16 [==============================] - 0s 2ms/step - loss: 0.0552 - accuracy: 0.9939 - val_loss: 1.4127 - val_accuracy:
0.7398
Epoch 90/100
16/16 [==============================] - 0s 2ms/step - loss: 0.0540 - accuracy: 0.9898 - val_loss: 1.3963 - val_accuracy:
0.7480
Epoch 91/100
16/16 [==============================] - 0s 2ms/step - loss: 0.0540 - accuracy: 0.9878 - val_loss: 1.4081 - val_accuracy:
0.7480
Epoch 92/100
16/16 [==============================] - 0s 3ms/step - loss: 0.0518 - accuracy: 0.9939 - val_loss: 1.4286 - val_accuracy:
0.7398
Epoch 93/100
16/16 [==============================] - 0s 3ms/step - loss: 0.0499 - accuracy: 0.9939 - val_loss: 1.4067 - val_accuracy:
0.7480
Epoch 94/100
16/16 [==============================] - 0s 3ms/step - loss: 0.0472 - accuracy: 0.9919 - val_loss: 1.4631 - val_accuracy:
0.7317
Epoch 95/100
16/16 [==============================] - 0s 2ms/step - loss: 0.0448 - accuracy: 0.9959 - val_loss: 1.4562 - val_accuracy:
0.7561
Epoch 96/100
16/16 [==============================] - 0s 2ms/step - loss: 0.0417 - accuracy: 0.9959 - val_loss: 1.5240 - val_accuracy:
0.7480
Epoch 97/100
16/16 [==============================] - 0s 3ms/step - loss: 0.0408 - accuracy: 0.9959 - val_loss: 1.4680 - val_accuracy:
0.7398
Epoch 98/100
16/16 [==============================] - 0s 2ms/step - loss: 0.0387 - accuracy: 0.9959 - val_loss: 1.5669 - val_accuracy:
0.7480
Epoch 99/100
16/16 [==============================] - 0s 3ms/step - loss: 0.0382 - accuracy: 0.9959 - val_loss: 1.5438 - val_accuracy:
0.7398
Epoch 100/100
16/16 [==============================] - 0s 3ms/step - loss: 0.0358 - accuracy: 0.9980 - val_loss: 1.5431 - val_accuracy:
0.7480
```

In [27]:

```
print(model.summary())
model.evaluate(X_test, y_test)
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                Output Shape              Param #
=================================================================
dense (Dense)               (None, 64)                576
_________________________________________________________________
```

```
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_2 (Dense)              (None, 16)                528
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 17
=================================================================
Total params: 3,201
Trainable params: 3,201
Non-trainable params: 0
_________________________________________________________________
None
5/5 [==============================] - 0s 1ms/step - loss: 1.4977 - accuracy: 0.6494
```

Out[27]:
```
[1.497668743133545, 0.649350643157959]
```

In [28]:
```python
from sklearn.metrics import confusion_matrix , classification_report
yp = model.predict(X_test)
y_pred = []
for element in yp:
    if element > 0.5:
        y_pred.append(1)
    else:
        y_pred.append(0)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.72      0.74      0.73        99
           1       0.51      0.49      0.50        55

    accuracy                           0.65       154
   macro avg       0.62      0.61      0.61       154
weighted avg       0.65      0.65      0.65       154
```

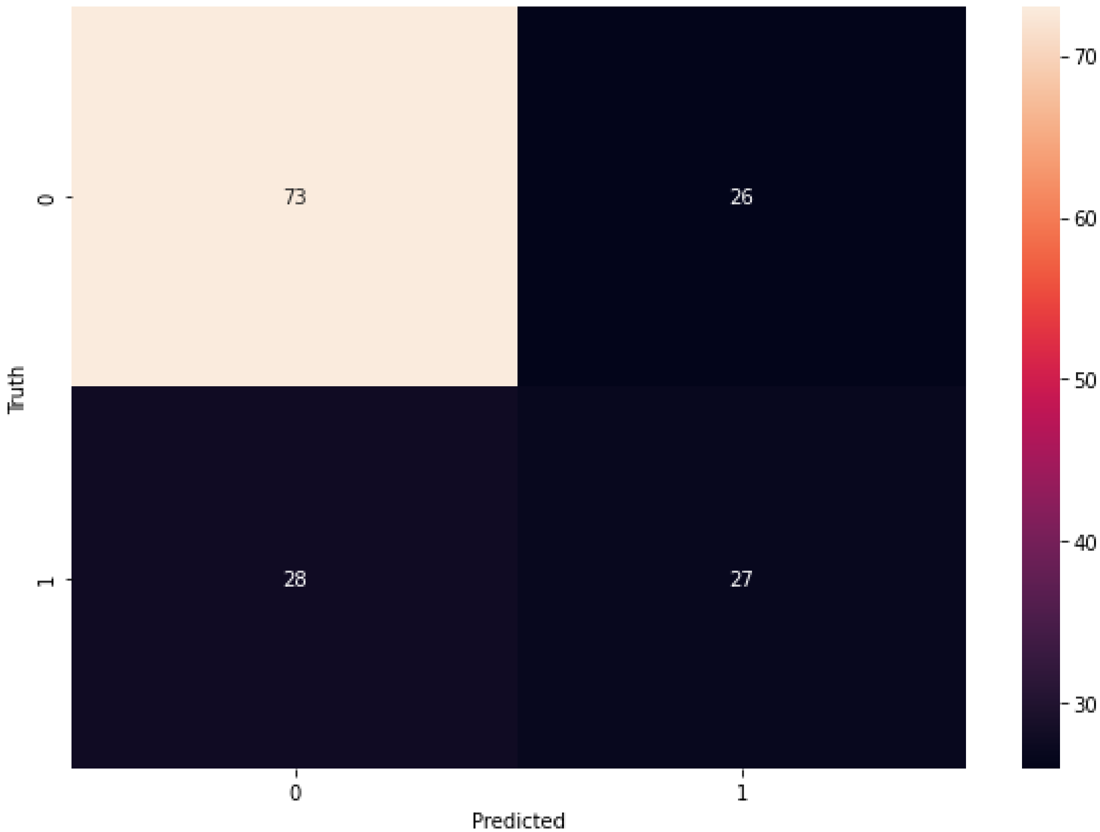
Precision – What percent of your predictions were correct? -> 76%

Recall – What percent of the positive cases did you catch? -> 73%

In [29]:
```python
import seaborn as sn
cm = tf.math.confusion_matrix(labels=y_test,predictions=y_pred)

plt.figure(figsize = (10,7))
sn.heatmap(cm, annot=True, fmt='d')
plt.xlabel('Predicted')
plt.ylabel('Truth')
```

Out[29]: Text(69.0, 0.5, 'Truth')

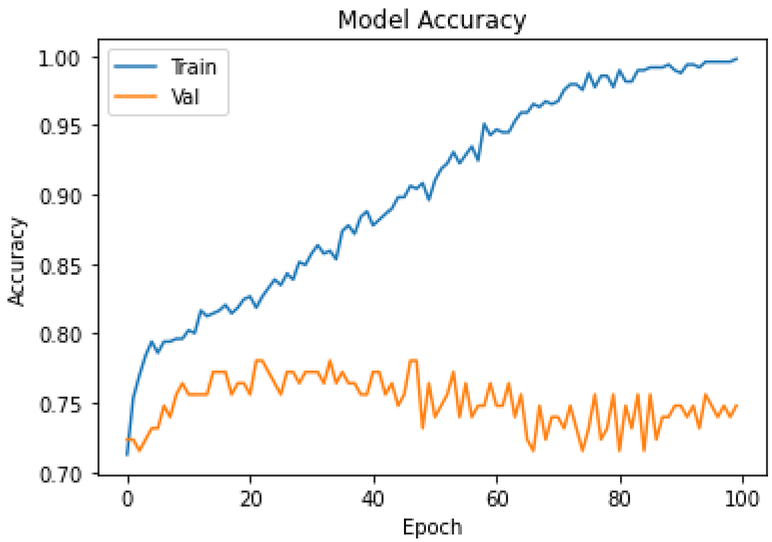


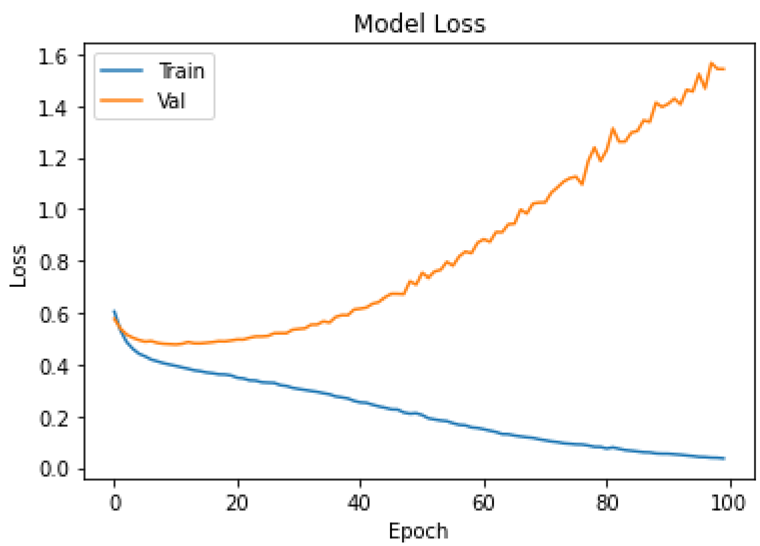
In [30]:

```
#accuracy
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend(['Train','Val'],loc='upper left')
plt.show()

#Loss
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model Loss')
plt.xlabel("Epoch")
```

```
plt.ylabel("Loss")
plt.legend(['Train','Val'],loc='upper left')
plt.show()
```





In [31]:
```
# Evaluating the model on the Test set
model.evaluate(X_test, y_test)
```

```
5/5 [==============================] - 0s 1ms/step - loss: 1.4977 - accuracy: 0.6494
```

Out[31]: [1.497668743133545, 0.649350643157959]

In [32]:
```
# Predicting
yp = model.predict(X_test)
yp[:5]
```

Out[32]:
```
array([[0.99979377],
       [0.00257984],
       [0.03728652],
       [0.0125863 ],
       [0.45886752]], dtype=float32)
```

Model Altering

In [33]:
```
#create a keras model
model2 = Sequential()
#once model object is created from Sequential we need layers to add
model2.add(Dense(12,input_dim=8,activation='relu'))
model2.add(Dense(12, activation='relu'))
model2.add(Dense(8,activation='relu'))
model2.add(Dense(4,activation='relu'))
model2.add(Dense(1,activation='sigmoid'))
```

In [34]:
```
#compile the model by providing parameters
model2.compile(loss='binary_crossentropy', optimizer='adam',metrics=['accuracy'])
```

In [35]:
```
model2.fit(X_train,y_train,epochs=250, batch_size=15)
```

```
Epoch 1/250
41/41 [==============================] - 0s 2ms/step - loss: 0.6840 - accuracy: 0.6498
Epoch 2/250
41/41 [==============================] - 0s 1ms/step - loss: 0.6619 - accuracy: 0.7101
Epoch 3/250
41/41 [==============================] - 0s 1ms/step - loss: 0.6238 - accuracy: 0.7280
Epoch 4/250
41/41 [==============================] - 0s 1ms/step - loss: 0.5675 - accuracy: 0.7443
Epoch 5/250
41/41 [==============================] - 0s 2ms/step - loss: 0.5242 - accuracy: 0.7443
Epoch 6/250
41/41 [==============================] - 0s 1ms/step - loss: 0.5015 - accuracy: 0.7606
Epoch 7/250
```

```
41/41 [==============================] - 0s 2ms/step - loss: 0.4876 - accuracy: 0.7638
Epoch 8/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4798 - accuracy: 0.7655
Epoch 9/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4716 - accuracy: 0.7671
Epoch 10/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4666 - accuracy: 0.7736
Epoch 11/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4584 - accuracy: 0.7704
Epoch 12/250
41/41 [==============================] - 0s 2ms/step - loss: 0.4548 - accuracy: 0.7769
Epoch 13/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4480 - accuracy: 0.7769
Epoch 14/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4445 - accuracy: 0.7818
Epoch 15/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4400 - accuracy: 0.7818
Epoch 16/250
41/41 [==============================] - 0s 2ms/step - loss: 0.4366 - accuracy: 0.7818
Epoch 17/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4326 - accuracy: 0.7818
Epoch 18/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4296 - accuracy: 0.7899
Epoch 19/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4259 - accuracy: 0.7850
Epoch 20/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4243 - accuracy: 0.7964
Epoch 21/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4213 - accuracy: 0.7883
Epoch 22/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4170 - accuracy: 0.7899
Epoch 23/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4158 - accuracy: 0.7997
Epoch 24/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4127 - accuracy: 0.8062
Epoch 25/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4116 - accuracy: 0.7997
Epoch 26/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4098 - accuracy: 0.8029
Epoch 27/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4096 - accuracy: 0.8062
Epoch 28/250
41/41 [==============================] - 0s 1ms/step - loss: 0.4078 - accuracy: 0.8013
Epoch 29/250
41/41 [==============================] - 0s 2ms/step - loss: 0.4045 - accuracy: 0.8062
Epoch 30/250
41/41 [==============================] - 0s 2ms/step - loss: 0.4020 - accuracy: 0.8013
Epoch 31/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3996 - accuracy: 0.8111
```

```
Epoch 32/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3984 - accuracy: 0.8094
Epoch 33/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3999 - accuracy: 0.8111
Epoch 34/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3975 - accuracy: 0.7997
Epoch 35/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3952 - accuracy: 0.8078
Epoch 36/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3937 - accuracy: 0.8078
Epoch 37/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3934 - accuracy: 0.8062
Epoch 38/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3927 - accuracy: 0.8111
Epoch 39/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3915 - accuracy: 0.8078
Epoch 40/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3898 - accuracy: 0.8046
Epoch 41/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3887 - accuracy: 0.8062
Epoch 42/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3876 - accuracy: 0.8062
Epoch 43/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3869 - accuracy: 0.8062
Epoch 44/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3869 - accuracy: 0.8078
Epoch 45/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3841 - accuracy: 0.8127
Epoch 46/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3830 - accuracy: 0.8111
Epoch 47/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3826 - accuracy: 0.8078
Epoch 48/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3809 - accuracy: 0.8078
Epoch 49/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3799 - accuracy: 0.8127
Epoch 50/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3783 - accuracy: 0.8094
Epoch 51/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3807 - accuracy: 0.8111
Epoch 52/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3792 - accuracy: 0.8127
Epoch 53/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3768 - accuracy: 0.8094
Epoch 54/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3765 - accuracy: 0.8094
Epoch 55/250
41/41 [==============================] - 0s 961us/step - loss: 0.3745 - accuracy: 0.8127
Epoch 56/250
```

```
41/41 [==============================] - 0s 1ms/step - loss: 0.3753 - accuracy: 0.8160
Epoch 57/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3721 - accuracy: 0.8143
Epoch 58/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3728 - accuracy: 0.8127
Epoch 59/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3722 - accuracy: 0.8143
Epoch 60/250
41/41 [==============================] - 0s 1000us/step - loss: 0.3703 - accuracy: 0.8143
Epoch 61/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3706 - accuracy: 0.8160
Epoch 62/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3685 - accuracy: 0.8192
Epoch 63/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3689 - accuracy: 0.8160
Epoch 64/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3672 - accuracy: 0.8241
Epoch 65/250
41/41 [==============================] - 0s 939us/step - loss: 0.3668 - accuracy: 0.8127
Epoch 66/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3628 - accuracy: 0.8274
Epoch 67/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3627 - accuracy: 0.8192
Epoch 68/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3631 - accuracy: 0.8176
Epoch 69/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3616 - accuracy: 0.8241
Epoch 70/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3607 - accuracy: 0.8241
Epoch 71/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3587 - accuracy: 0.8225
Epoch 72/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3593 - accuracy: 0.8208
Epoch 73/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3585 - accuracy: 0.8192
Epoch 74/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3556 - accuracy: 0.8208
Epoch 75/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3568 - accuracy: 0.8208
Epoch 76/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3562 - accuracy: 0.8257
Epoch 77/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3524 - accuracy: 0.8290
Epoch 78/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3533 - accuracy: 0.8241
Epoch 79/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3537 - accuracy: 0.8225
Epoch 80/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3516 - accuracy: 0.8290
```

```
Epoch 81/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3525 - accuracy: 0.8339
Epoch 82/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3528 - accuracy: 0.8225
Epoch 83/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3482 - accuracy: 0.8371
Epoch 84/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3478 - accuracy: 0.8306
Epoch 85/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3449 - accuracy: 0.8404
Epoch 86/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3462 - accuracy: 0.8436
Epoch 87/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3464 - accuracy: 0.8306
Epoch 88/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3444 - accuracy: 0.8322
Epoch 89/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3440 - accuracy: 0.8257
Epoch 90/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3433 - accuracy: 0.8388
Epoch 91/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3435 - accuracy: 0.8388
Epoch 92/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3410 - accuracy: 0.8420
Epoch 93/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3434 - accuracy: 0.8355
Epoch 94/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3393 - accuracy: 0.8355
Epoch 95/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3412 - accuracy: 0.8388
Epoch 96/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3392 - accuracy: 0.8404
Epoch 97/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3390 - accuracy: 0.8485
Epoch 98/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3382 - accuracy: 0.8404
Epoch 99/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3355 - accuracy: 0.8453
Epoch 100/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3366 - accuracy: 0.8453
Epoch 101/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3358 - accuracy: 0.8404
Epoch 102/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3354 - accuracy: 0.8453
Epoch 103/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3350 - accuracy: 0.8436
Epoch 104/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3347 - accuracy: 0.8371
Epoch 105/250
```

```
41/41 [==============================] - 0s 1ms/step - loss: 0.3362 - accuracy: 0.8371
Epoch 106/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3324 - accuracy: 0.8436
Epoch 107/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3319 - accuracy: 0.8388
Epoch 108/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3303 - accuracy: 0.8436
Epoch 109/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3306 - accuracy: 0.8453
Epoch 110/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3320 - accuracy: 0.8436
Epoch 111/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3298 - accuracy: 0.8453
Epoch 112/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3287 - accuracy: 0.8534
Epoch 113/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3276 - accuracy: 0.8502
Epoch 114/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3269 - accuracy: 0.8567
Epoch 115/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3268 - accuracy: 0.8420
Epoch 116/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3284 - accuracy: 0.8388
Epoch 117/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3272 - accuracy: 0.8436
Epoch 118/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3300 - accuracy: 0.8404
Epoch 119/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3258 - accuracy: 0.8534
Epoch 120/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3260 - accuracy: 0.8534
Epoch 121/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3241 - accuracy: 0.8485
Epoch 122/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3240 - accuracy: 0.8518
Epoch 123/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3231 - accuracy: 0.8485
Epoch 124/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3223 - accuracy: 0.8453
Epoch 125/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3232 - accuracy: 0.8469
Epoch 126/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3206 - accuracy: 0.8616
Epoch 127/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3192 - accuracy: 0.8567
Epoch 128/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3187 - accuracy: 0.8518
Epoch 129/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3173 - accuracy: 0.8502
```

```
Epoch 130/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3161 - accuracy: 0.8648
Epoch 131/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3158 - accuracy: 0.8518
Epoch 132/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3186 - accuracy: 0.8469
Epoch 133/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3150 - accuracy: 0.8583
Epoch 134/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3154 - accuracy: 0.8567
Epoch 135/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3146 - accuracy: 0.8550
Epoch 136/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3150 - accuracy: 0.8518
Epoch 137/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3137 - accuracy: 0.8616
Epoch 138/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3134 - accuracy: 0.8599
Epoch 139/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3130 - accuracy: 0.8550
Epoch 140/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3116 - accuracy: 0.8616
Epoch 141/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3111 - accuracy: 0.8616
Epoch 142/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3110 - accuracy: 0.8550
Epoch 143/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3112 - accuracy: 0.8583
Epoch 144/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3085 - accuracy: 0.8567
Epoch 145/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3096 - accuracy: 0.8583
Epoch 146/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3078 - accuracy: 0.8616
Epoch 147/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3061 - accuracy: 0.8599
Epoch 148/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3066 - accuracy: 0.8664
Epoch 149/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3086 - accuracy: 0.8616
Epoch 150/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3080 - accuracy: 0.8599
Epoch 151/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3113 - accuracy: 0.8567
Epoch 152/250
41/41 [==============================] - 0s 950us/step - loss: 0.3057 - accuracy: 0.8648
Epoch 153/250
41/41 [==============================] - 0s 971us/step - loss: 0.3036 - accuracy: 0.8599
Epoch 154/250
```

```
41/41 [==============================] - 0s 1ms/step - loss: 0.3005 - accuracy: 0.8616
Epoch 155/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3026 - accuracy: 0.8664
Epoch 156/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3040 - accuracy: 0.8648
Epoch 157/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3029 - accuracy: 0.8664
Epoch 158/250
41/41 [==============================] - 0s 2ms/step - loss: 0.3015 - accuracy: 0.8713
Epoch 159/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3003 - accuracy: 0.8697
Epoch 160/250
41/41 [==============================] - 0s 1ms/step - loss: 0.3001 - accuracy: 0.8730
Epoch 161/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2966 - accuracy: 0.8681
Epoch 162/250
41/41 [==============================] - ETA: 0s - loss: 0.3963 - accuracy: 0.80 - 0s 1ms/step - loss: 0.3014 - accuracy:
0.8583
Epoch 163/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2969 - accuracy: 0.8697
Epoch 164/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2952 - accuracy: 0.8697
Epoch 165/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2954 - accuracy: 0.8713
Epoch 166/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2953 - accuracy: 0.8746
Epoch 167/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2925 - accuracy: 0.8713
Epoch 168/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2919 - accuracy: 0.8730
Epoch 169/250
41/41 [==============================] - 0s 2ms/step - loss: 0.2933 - accuracy: 0.8713
Epoch 170/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2914 - accuracy: 0.8779
Epoch 171/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2906 - accuracy: 0.8746
Epoch 172/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2895 - accuracy: 0.8713
Epoch 173/250
41/41 [==============================] - 0s 2ms/step - loss: 0.2919 - accuracy: 0.8697
Epoch 174/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2931 - accuracy: 0.8730
Epoch 175/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2900 - accuracy: 0.8730
Epoch 176/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2885 - accuracy: 0.8681
Epoch 177/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2911 - accuracy: 0.8681
Epoch 178/250
```

```
41/41 [==============================] - 0s 1ms/step - loss: 0.2840 - accuracy: 0.8713
Epoch 179/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2832 - accuracy: 0.8730
Epoch 180/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2825 - accuracy: 0.8713
Epoch 181/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2880 - accuracy: 0.8664
Epoch 182/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2835 - accuracy: 0.8779
Epoch 183/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2849 - accuracy: 0.8795
Epoch 184/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2825 - accuracy: 0.8762
Epoch 185/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2782 - accuracy: 0.8762
Epoch 186/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2785 - accuracy: 0.8795
Epoch 187/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2782 - accuracy: 0.8762
Epoch 188/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2764 - accuracy: 0.8844
Epoch 189/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2774 - accuracy: 0.8762
Epoch 190/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2788 - accuracy: 0.8811
Epoch 191/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2792 - accuracy: 0.8779
Epoch 192/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2729 - accuracy: 0.8795
Epoch 193/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2719 - accuracy: 0.8876
Epoch 194/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2739 - accuracy: 0.8827
Epoch 195/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2720 - accuracy: 0.8860
Epoch 196/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2701 - accuracy: 0.8795
Epoch 197/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2725 - accuracy: 0.8795
Epoch 198/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2699 - accuracy: 0.8860
Epoch 199/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2709 - accuracy: 0.8860
Epoch 200/250
41/41 [==============================] - 0s 981us/step - loss: 0.2658 - accuracy: 0.8827
Epoch 201/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2694 - accuracy: 0.8779
Epoch 202/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2673 - accuracy: 0.8876
```

```
Epoch 203/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2683 - accuracy: 0.8811
Epoch 204/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2673 - accuracy: 0.8860
Epoch 205/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2632 - accuracy: 0.8811
Epoch 206/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2620 - accuracy: 0.8893
Epoch 207/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2668 - accuracy: 0.8779
Epoch 208/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2621 - accuracy: 0.8860
Epoch 209/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2610 - accuracy: 0.8909
Epoch 210/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2587 - accuracy: 0.8909
Epoch 211/250
41/41 [==============================] - 0s 964us/step - loss: 0.2595 - accuracy: 0.8893
Epoch 212/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2576 - accuracy: 0.8876
Epoch 213/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2580 - accuracy: 0.8860
Epoch 214/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2571 - accuracy: 0.8844
Epoch 215/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2561 - accuracy: 0.8827
Epoch 216/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2572 - accuracy: 0.8876
Epoch 217/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2585 - accuracy: 0.8925
Epoch 218/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2541 - accuracy: 0.8925
Epoch 219/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2531 - accuracy: 0.8909
Epoch 220/250
41/41 [==============================] - 0s 2ms/step - loss: 0.2519 - accuracy: 0.8876
Epoch 221/250
41/41 [==============================] - 1s 19ms/step - loss: 0.2558 - accuracy: 0.8827 0s - loss: 0.2849 - accuracy: 0.8
6 - ETA: 0s - loss: 0.282
Epoch 222/250
41/41 [==============================] - 0s 3ms/step - loss: 0.2500 - accuracy: 0.8876
Epoch 223/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2506 - accuracy: 0.8876
Epoch 224/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2489 - accuracy: 0.8990
Epoch 225/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2508 - accuracy: 0.8941
Epoch 226/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2512 - accuracy: 0.8876
```

```
Epoch 227/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2505 - accuracy: 0.8990: 0s - loss: 0.2328 - accuracy: 0.9
0
Epoch 228/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2482 - accuracy: 0.8893
Epoch 229/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2426 - accuracy: 0.8974
Epoch 230/250
41/41 [==============================] - 0s 2ms/step - loss: 0.2475 - accuracy: 0.8941
Epoch 231/250
41/41 [==============================] - 0s 2ms/step - loss: 0.2442 - accuracy: 0.8909
Epoch 232/250
41/41 [==============================] - 0s 2ms/step - loss: 0.2431 - accuracy: 0.8925
Epoch 233/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2431 - accuracy: 0.8958
Epoch 234/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2422 - accuracy: 0.8941
Epoch 235/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2381 - accuracy: 0.8990
Epoch 236/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2374 - accuracy: 0.8990
Epoch 237/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2386 - accuracy: 0.8958
Epoch 238/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2362 - accuracy: 0.8990
Epoch 239/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2385 - accuracy: 0.8974
Epoch 240/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2357 - accuracy: 0.9007
Epoch 241/250
41/41 [==============================] - 0s 2ms/step - loss: 0.2348 - accuracy: 0.9072
Epoch 242/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2383 - accuracy: 0.8990
Epoch 243/250
41/41 [==============================] - 0s 2ms/step - loss: 0.2338 - accuracy: 0.9007
Epoch 244/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2301 - accuracy: 0.9023
Epoch 245/250
41/41 [==============================] - 0s 2ms/step - loss: 0.2356 - accuracy: 0.8958
Epoch 246/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2315 - accuracy: 0.9023
Epoch 247/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2344 - accuracy: 0.8941
Epoch 248/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2338 - accuracy: 0.8974
Epoch 249/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2280 - accuracy: 0.8990
Epoch 250/250
41/41 [==============================] - 0s 1ms/step - loss: 0.2292 - accuracy: 0.9007
```

Out[35]: <tensorflow.python.keras.callbacks.History at 0x1d75d231820>

In [36]:

```
print(model2.summary())
model2.evaluate(X_test, y_test)
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_4 (Dense)              (None, 12)                108
_________________________________________________________________
dense_5 (Dense)              (None, 12)                156
_________________________________________________________________
dense_6 (Dense)              (None, 8)                 104
_________________________________________________________________
dense_7 (Dense)              (None, 4)                 36
_________________________________________________________________
dense_8 (Dense)              (None, 1)                 5
=================================================================
Total params: 409
Trainable params: 409
Non-trainable params: 0
_________________________________________________________________
None
5/5 [==============================] - 0s 2ms/step - loss: 0.9631 - accuracy: 0.6429
```

Out[36]: [0.9631482362747192, 0.6428571343421936]

In [37]:

```
from sklearn.metrics import confusion_matrix , classification_report
yp = model2.predict(X_test)
y_pred = []
for element in yp:
    if element > 0.5:
        y_pred.append(1)
    else:
        y_pred.append(0)
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.73      0.70      0.72        99
           1       0.50      0.55      0.52        55

    accuracy                           0.64       154
   macro avg       0.62      0.62      0.62       154
```

```
weighted avg       0.65      0.64      0.65       154
```

Training the first model resulted in a bit higher accuracy. However, the classification report metrics are better after altering the model. I can notice that the precision and the recall are higher than they were.

In [38]:

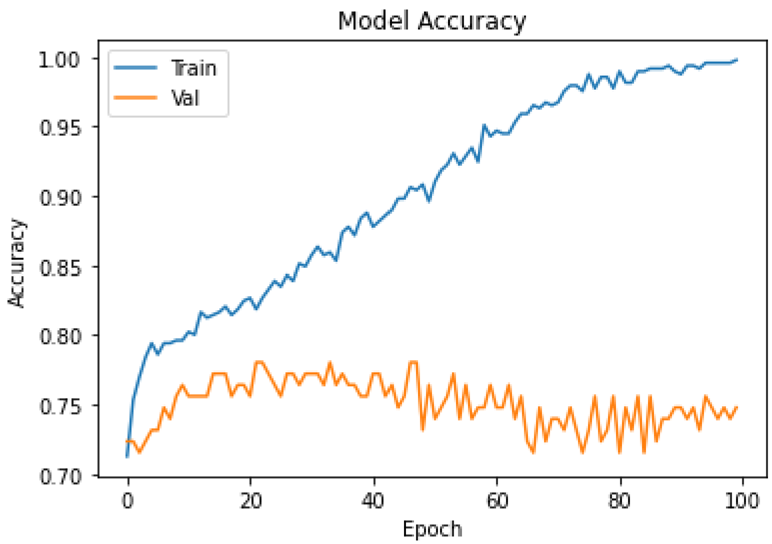
```
print(hist.history.keys())
```

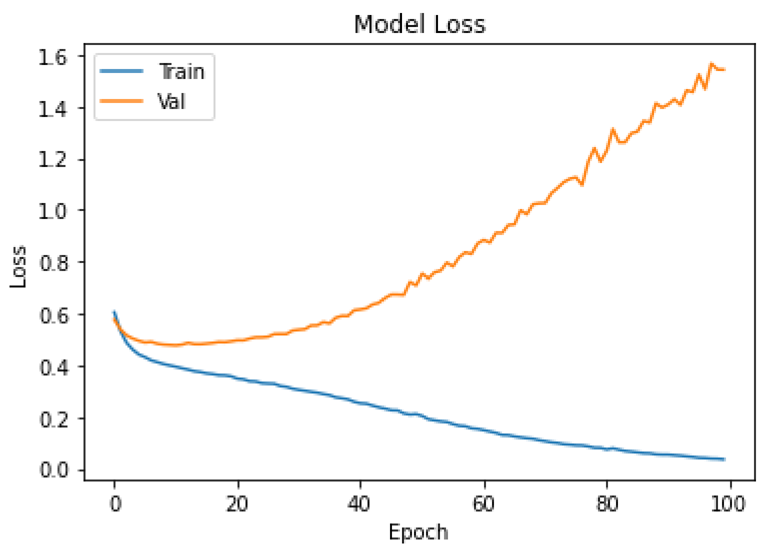
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [39]:

```
#accuracy
plt.plot(hist.history['accuracy'])
plt.plot(hist.history['val_accuracy'])
plt.title('Model Accuracy')
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend(['Train','Val'],loc='upper left')
plt.show()

#Loss
plt.plot(hist.history['loss'])
plt.plot(hist.history['val_loss'])
plt.title('Model Loss')
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.legend(['Train','Val'],loc='upper left')
plt.show()
```





In [40]:

```
# Evaluating the model on the Test set
model2.evaluate(X_test, y_test)
```

```
5/5 [==============================] - 0s 1ms/step - loss: 0.9631 - accuracy: 0.6429
```

Out[40]:

```
[0.9631482362747192, 0.6428571343421936]
```

In [41]:

```
# Predicting
yp = model2.predict(X_test)
```

```
yp[:5]
```

Out[41]:
```
array([[0.9331449 ],
       [0.00583637],
       [0.00678438],
       [0.50993264],
       [0.13698545]], dtype=float32)
```

Conclusion

Using the Artificial Neural Networks(ANN) model can design and implement the complex medical processes by software. The software systems are more effective and efficient in various medical fields including predicting, diagnosing, treating and helping the surgeons and physicians, and the general population. These systems can be implemented in a parallel way and are distributed in different scales. In general, the Artificial Neural Networks are parallel processing systems that are used to detect complex patterns in the data. The aim of this notebook was to determine effective variables and their impact on diabetes and estimating whether a neural network model can predict diabetes. The results of training the model are not that high for now, as it could be seen on the above figures. The first model consists of 4 layers. The activation function chosen was ReLU for the first layer, ReLU for the two hidden layers and sigmoid for the output layer. The **"Altering model"** was with 5 layers. The input layer was a ReLU and the other 3 hidden layers were ReLU and the output layer was a segmoid.

References

- https://www.youtube.com/watch?v=BYY8eSJkJtA&ab_channel=LearnPythonwithRune
- https://link.springer.com/chapter/10.1007/978-981-13-1642-5_59#:~:text=Among%20several%20algorithms%20of%20Machine,with%20the%20sample%20test%20data.
- https://www.researchgate.net/publication/329829533_Diabetes_Prediction_Using_Artificial_Neural_Network
- https://www.hindawi.com/journals/complexity/2021/5525271/