In [2]:
```
#Given a dataset of various metrics can we predict if a patient has diabetes
# Import libraries
import pandas as pd
import numpy as np
import tensorflow as tf
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
```

In [3]:
```
# Read the data
data = pd.read_csv('./files/diabetes.csv')
data.head()
```

Out[3]:

	Number of times pregnant	Plasma glucose concentration	Diastolic blood pressure	Triceps skin fold thickness	2-Hour serum insulin	Body mass index	Diabetes pedigree function	Age	Class variable
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
4	0	137	40	35	168	43.1	2.288	33	1

In [4]:
```
#Check for data quality
data.isna().sum()
```

Out[4]:
```
Number of times pregnant        0
Plasma glucose concentration    0
Diastolic blood pressure        0
Triceps skin fold thickness     0
2-Hour serum insulin            0
Body mass index                 0
Diabetes pedigree function      0
Age                             0
Class variable                  0
dtype: int64
```

In [5]:
```
data.dtypes
```

Out[5]:
```
Number of times pregnant          int64
Plasma glucose concentration      int64
Diastolic blood pressure          int64
Triceps skin fold thickness       int64
2-Hour serum insulin              int64
Body mass index                 float64
Diabetes pedigree function      float64
Age                               int64
Class variable                    int64
dtype: object
```

In [6]:
```
# Create dataset
X = data.iloc[:,:-1]
y = data.iloc[:, -1]
```

In [7]:
```
#Create training and test set
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

In [8]:
```
#Calculate average accuracy for 10 runs
accuracies = []

for i in range(10):
  tf.random.set_seed(i)
  model = Sequential()
  model.add(Dense(1, input_dim=8, activation='sigmoid'))
  model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
  model.fit(X_train, y_train, epochs=1500, batch_size=100, verbose=0)
  _, accuracy = model.evaluate(X_test, y_test)
  accuracies.append(accuracy*100)
```

```
2022-04-23 17:55:55.990859: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:939] successful NUMA node read
from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-04-23 17:55:56.040391: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:939] successful NUMA node read
from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-04-23 17:55:56.040975: I tensorflow/stream_executor/cuda/cuda_gpu_executor.cc:939] successful NUMA node read
from SysFS had negative value (-1), but there must be at least one NUMA node, so returning NUMA node zero
2022-04-23 17:55:56.041420: I tensorflow/core/common_runtime/gpu/gpu_device.cc:1900] Ignoring visible gpu device
(device: 0, name: Quadro K1000M, pci bus id: 0000:01:00.0, compute capability: 3.0) with Cuda compute capability
3.0. The minimum required Cuda capability is 3.5.
2022-04-23 17:55:56.042651: I tensorflow/core/platform/cpu_feature_guard.cc:151] This TensorFlow binary is optimi
zed with oneAPI Deep Neural Network Library (oneDNN) to use the following CPU instructions in performance-critica
l operations:  SSE4.1 SSE4.2 AVX
To enable them in other operations, rebuild TensorFlow with the appropriate compiler flags.
6/6 [==============================] - 0s 2ms/step - loss: 0.5216 - accuracy: 0.7344
6/6 [==============================] - 0s 2ms/step - loss: 0.5231 - accuracy: 0.7552
6/6 [==============================] - 0s 1ms/step - loss: 0.5192 - accuracy: 0.7604
6/6 [==============================] - 0s 2ms/step - loss: 0.5193 - accuracy: 0.7500
6/6 [==============================] - 0s 1ms/step - loss: 0.5285 - accuracy: 0.7552
6/6 [==============================] - 0s 2ms/step - loss: 0.5209 - accuracy: 0.7396
6/6 [==============================] - 0s 1ms/step - loss: 0.5206 - accuracy: 0.7448
6/6 [==============================] - 0s 2ms/step - loss: 0.5289 - accuracy: 0.7552
6/6 [==============================] - 0s 1ms/step - loss: 0.5219 - accuracy: 0.7448
6/6 [==============================] - 0s 1ms/step - loss: 0.5238 - accuracy: 0.7760
```

In [9]:
```
sum(accuracies) / len(accuracies)
```

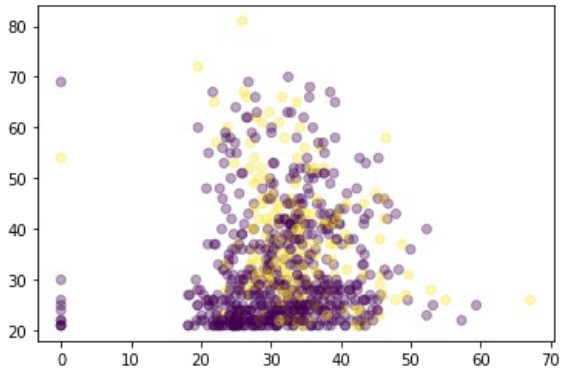
Out[9]:
```
75.15625
```

In [10]:
```
#Predict values
y_pred = model.predict(X)
y_pred = np.where(y_pred < .5, 0, 1)
```

In [11]:
```
#Visualize correct vs incorrect predictions
differ = np.abs(y.to_numpy() - y_pred.T)

fig, ax = plt.subplots()
ax.scatter(x=X['Body mass index'], y=X['Age'], c = differ, alpha = .35)
```

Out[11]:
```
<matplotlib.collections.PathCollection at 0x7fea3d0d1b20>
```



In [ ]:
```
```