Neural Network to Predict College Admission Using Tensorflow

In [1]:
```python
# import libraries and data
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import os
from jupyterthemes import jtplot
jtplot.style(theme='monokai', context='notebook', ticks=True, grid=False)
```

In [2]:
```python
# read csv
admission_df = pd.read_csv('Admission_Predict.csv')
admission_df.drop('Serial No.', axis = 1, inplace = True)
```

In [3]:
```python
# check for null values
admission_df.isnull().sum()
```

Out[3]:
```
GRE Score            0
TOEFL Score          0
University Rating    0
SOP                  0
LOR                  0
CGPA                 0
Research             0
Chance of Admit      0
dtype: int64
```

In [4]:
```python
# Group by university ranking
df_university = admission_df.groupby(by = 'University Rating').mean()
df_university
```

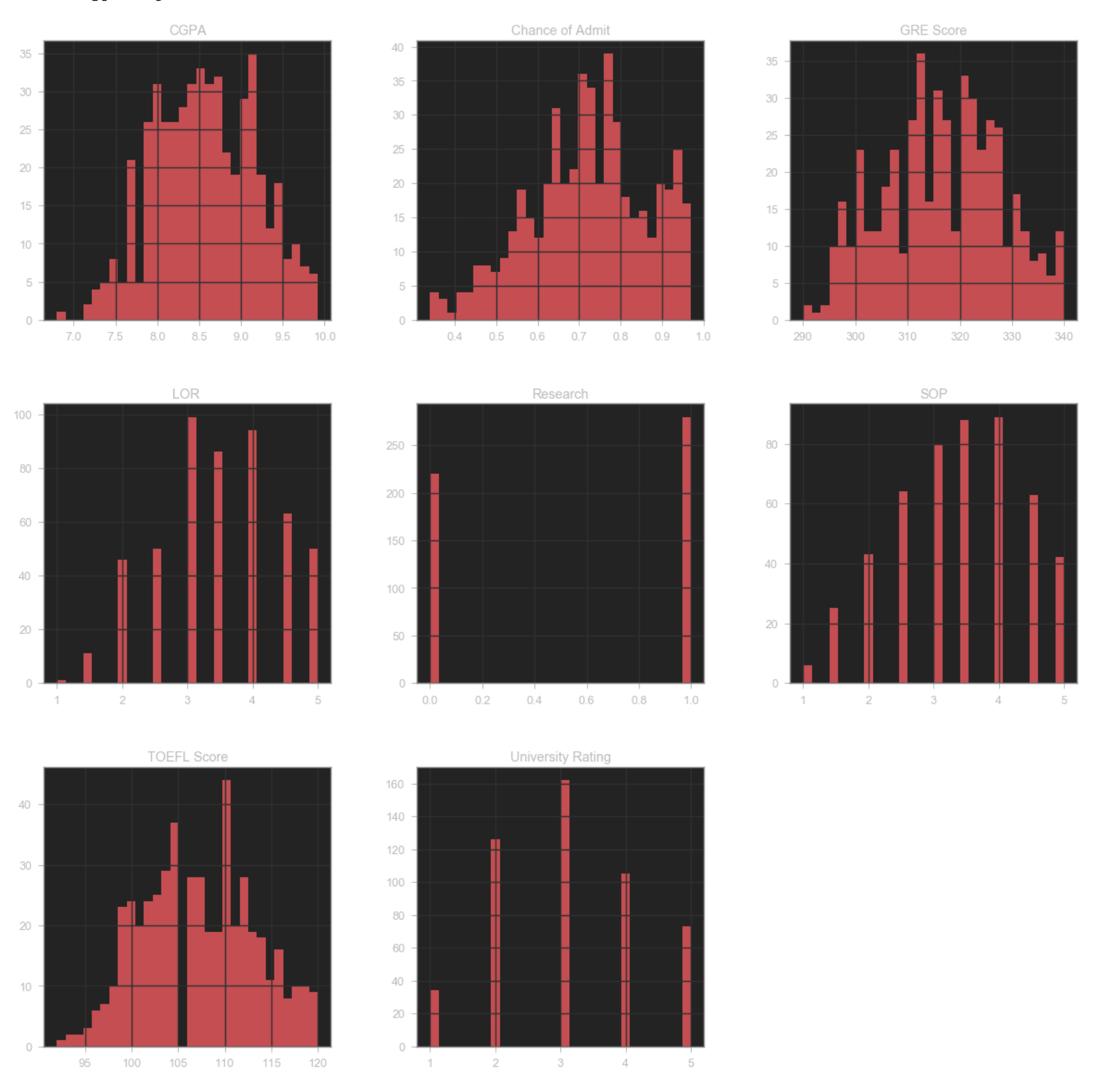
Out[4]:

University Rating	GRE Score	TOEFL Score	SOP	LOR	CGPA	Research	Chance of Admit
1	304.911765	100.205882	1.941176	2.426471	7.798529	0.294118	0.562059
2	309.134921	103.444444	2.682540	2.956349	8.177778	0.293651	0.626111
3	315.030864	106.314815	3.308642	3.401235	8.500123	0.537037	0.702901
4	323.304762	110.961905	4.000000	3.947619	8.936667	0.780952	0.801619
5	327.890411	113.438356	4.479452	4.404110	9.278082	0.876712	0.888082

Visualize Data

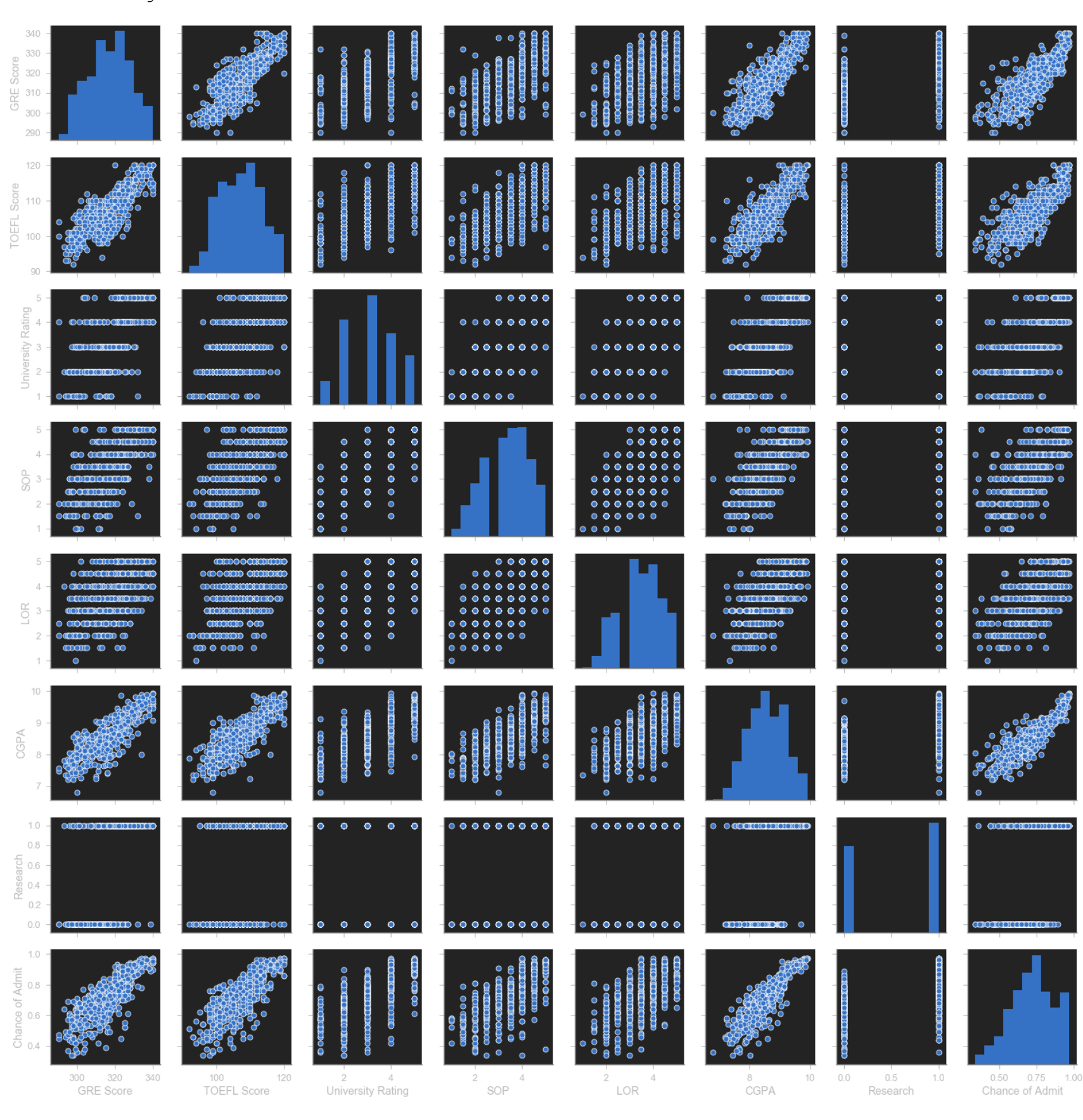
In [5]:
```python
# view histograms
admission_df.hist(bins = 30, figsize = (20, 20), color = 'r')
```

Out[5]:
```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x000002DAB8FAABC8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002DAB7807DC8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002DAB90322C8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002DAB9066F48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002DAB90A2088>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002DAB90DA188>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x000002DAB9111288>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002DAB914A388>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x000002DAB914FF48>]],
      dtype=object)
```

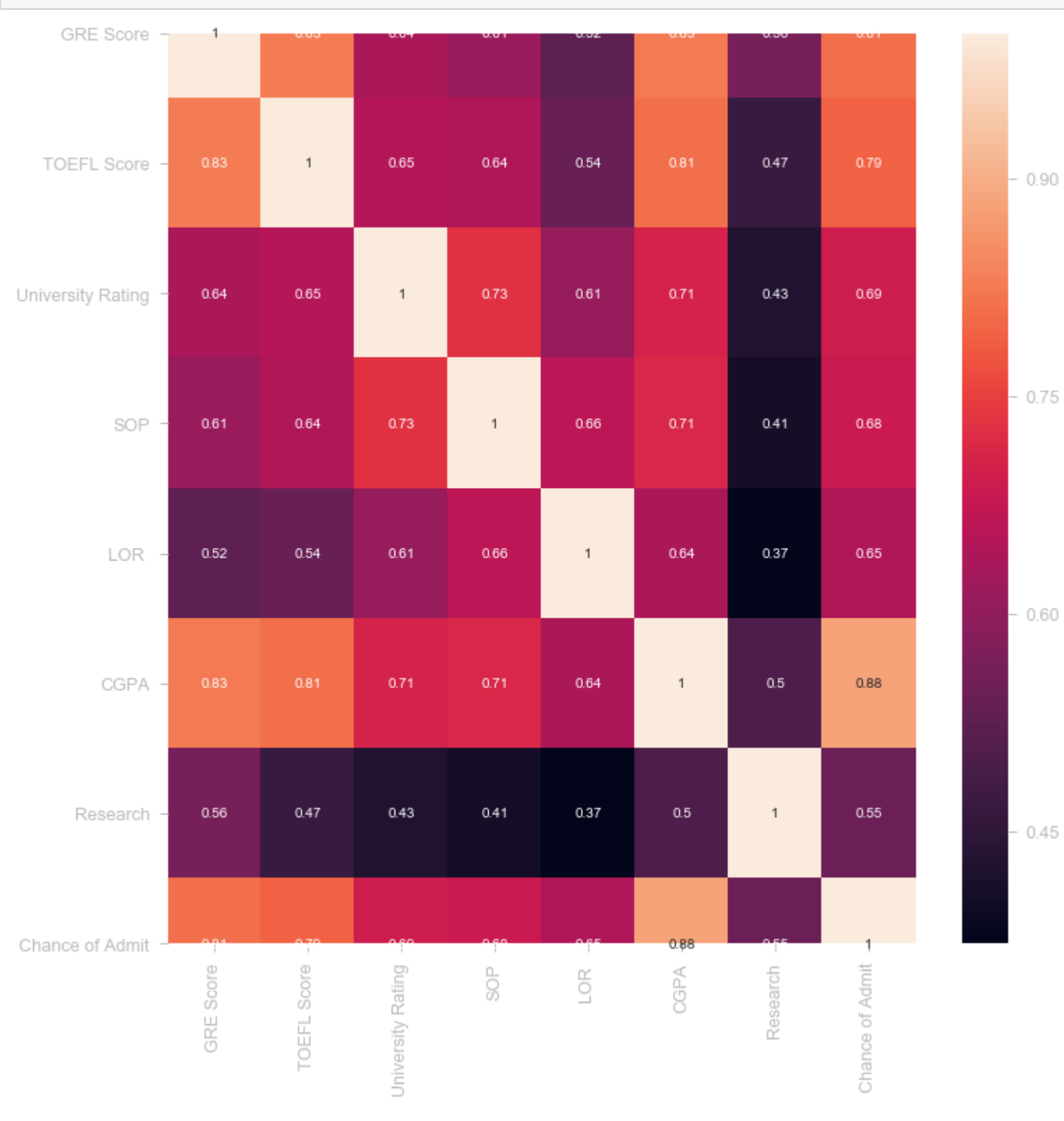


In [6]:
```python
# view pairplot
sns.pairplot(admission_df)
```

Out[6]: `<seaborn.axisgrid.PairGrid at 0x2dab9da2b88>`



In [7]:
```python
# plot correlations
corr_matrix = admission_df.corr()
plt.figure(figsize = (12, 12))
sns.heatmap(corr_matrix, annot = True)
plt.show()
```



Create Training & Testing Datasets

In [8]:
```python
X = admission_df.drop(columns = ['Chance of Admit'])
```

In [9]:
```python
y = admission_df['Chance of Admit']
```

In [10]:
```python
X = np.array(X)
y = np.array(y)
```

In [11]:
```python
# reshape y
y = y.reshape(-1,1)
y.shape
```

Out[11]: `(500, 1)`

In [12]:
```python
# scale data before training model
# remove bias by creating consistent range for each feature

from sklearn.preprocessing import StandardScaler, MinMaxScaler

scaler_x = StandardScaler()
X = scaler_x.fit_transform(X)

# same for y
scaler_y = StandardScaler()
y = scaler_y.fit_transform(y)
```

In [13]:
```python
# spliting the data
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.15)
```

In [14]:
```python
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, accuracy_score

LinearRegression_model = LinearRegression()
LinearRegression_model.fit(X_train, y_train)
```

Out[14]: `LinearRegression()`

In [15]:
```python
accuracy_LinearRegression = LinearRegression_model.score(X_test, y_test)
accuracy_LinearRegression
```

Out[15]: `0.8421191140683675`

Neural Network

In [16]:
```python
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.optimizers import Adam
```

In [17]:
```python
# Dropout ensures we are not overfitting
# relu is rectified linear unit
# build network sequentially
# first, add layer to model that is dense (fully connected), has 50 neurons and 7 input dimensions

ANN_model = keras.Sequential()
ANN_model.add(Dense(50, input_dim = 7))
ANN_model.add(Activation('relu'))

ANN_model.add(Dense(150))
ANN_model.add(Activation('relu'))
ANN_model.add(Dropout(0.5))

ANN_model.add(Dense(150))
ANN_model.add(Activation('relu'))
ANN_model.add(Dropout(0.5))

# implement regression task
ANN_model.add(Dense(50))
ANN_model.add(Activation('linear'))
ANN_model.add(Dense(1))

ANN_model.compile(loss = 'mse', optimizer = 'adam')
ANN_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 50)                400
_________________________________________________________________
activation (Activation)      (None, 50)                0
_________________________________________________________________
dense_1 (Dense)              (None, 150)               7650
_________________________________________________________________
activation_1 (Activation)    (None, 150)               0
_________________________________________________________________
dropout (Dropout)            (None, 150)               0
_________________________________________________________________
dense_2 (Dense)              (None, 150)               22650
_________________________________________________________________
activation_2 (Activation)    (None, 150)               0
_________________________________________________________________
dropout_1 (Dropout)          (None, 150)               0
_________________________________________________________________
dense_3 (Dense)              (None, 50)                7550
_________________________________________________________________
activation_3 (Activation)    (None, 50)                0
_________________________________________________________________
dense_4 (Dense)              (None, 1)                 51
=================================================================
Total params: 38,301
Trainable params: 38,301
Non-trainable params: 0
_________________________________________________________________
```

In [18]:
```python
# compile and specify loss
ANN_model.compile(optimizer='Adam', loss='mean_squared_error')
```

In [19]:
```python
epochs_hist = ANN_model.fit(X_train, y_train, epochs = 100, batch_size = 20, validation_split = 0.2)
```

```
Epoch 1/100
17/17 [==============================] - 0s 12ms/step - loss: 0.6400 - val_loss: 0.2462
Epoch 2/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2999 - val_loss: 0.2824
Epoch 3/100
17/17 [==============================] - 0s 3ms/step - loss: 0.3037 - val_loss: 0.2459
Epoch 4/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2818 - val_loss: 0.2379
Epoch 5/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2682 - val_loss: 0.2533
Epoch 6/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2380 - val_loss: 0.2472
Epoch 7/100
17/17 [==============================] - 0s 4ms/step - loss: 0.2587 - val_loss: 0.2517
Epoch 8/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2456 - val_loss: 0.2481
Epoch 9/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2565 - val_loss: 0.2628
Epoch 10/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2572 - val_loss: 0.2601
Epoch 11/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2272 - val_loss: 0.2797
Epoch 12/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2087 - val_loss: 0.2570
Epoch 13/100
17/17 [==============================] - 0s 4ms/step - loss: 0.2287 - val_loss: 0.2476
Epoch 14/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1924 - val_loss: 0.2593
Epoch 15/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2120 - val_loss: 0.2715
Epoch 16/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2079 - val_loss: 0.2433
Epoch 17/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2071 - val_loss: 0.2547
Epoch 18/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1957 - val_loss: 0.2560
Epoch 19/100
17/17 [==============================] - 0s 4ms/step - loss: 0.2092 - val_loss: 0.2584
Epoch 20/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1823 - val_loss: 0.2664
Epoch 21/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1911 - val_loss: 0.2730
Epoch 22/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1917 - val_loss: 0.2643
Epoch 23/100
17/17 [==============================] - 0s 3ms/step - loss: 0.2027 - val_loss: 0.3051
Epoch 24/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1926 - val_loss: 0.2543
Epoch 25/100
17/17 [==============================] - 0s 6ms/step - loss: 0.1906 - val_loss: 0.3089
Epoch 26/100
17/17 [==============================] - 0s 6ms/step - loss: 0.2015 - val_loss: 0.2866
Epoch 27/100
17/17 [==============================] - 0s 6ms/step - loss: 0.1918 - val_loss: 0.2680
Epoch 28/100
17/17 [==============================] - 0s 6ms/step - loss: 0.1760 - val_loss: 0.2817
Epoch 29/100
17/17 [==============================] - 0s 6ms/step - loss: 0.2070 - val_loss: 0.3094
Epoch 30/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1784 - val_loss: 0.2685
Epoch 31/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1922 - val_loss: 0.2752
Epoch 32/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1826 - val_loss: 0.2830
Epoch 33/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1942 - val_loss: 0.3076
Epoch 34/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1630 - val_loss: 0.2570
Epoch 35/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1903 - val_loss: 0.2856
Epoch 36/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1673 - val_loss: 0.2856
Epoch 37/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1580 - val_loss: 0.2748
Epoch 38/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1737 - val_loss: 0.2866
Epoch 39/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1672 - val_loss: 0.2570
Epoch 40/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1685 - val_loss: 0.2653
Epoch 41/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1651 - val_loss: 0.2937
Epoch 42/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1568 - val_loss: 0.2719
Epoch 43/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1646 - val_loss: 0.2769
Epoch 44/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1440 - val_loss: 0.2828
Epoch 45/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1774 - val_loss: 0.3035
Epoch 46/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1681 - val_loss: 0.2632
Epoch 47/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1521 - val_loss: 0.3071
Epoch 48/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1523 - val_loss: 0.2576
Epoch 49/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1600 - val_loss: 0.2574
Epoch 50/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1728 - val_loss: 0.2870
Epoch 51/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1646 - val_loss: 0.2515
Epoch 52/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1487 - val_loss: 0.2670
Epoch 53/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1591 - val_loss: 0.2630
Epoch 54/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1415 - val_loss: 0.2801
Epoch 55/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1496 - val_loss: 0.2928
Epoch 56/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1368 - val_loss: 0.2887
Epoch 57/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1470 - val_loss: 0.2764
Epoch 58/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1467 - val_loss: 0.2875
Epoch 59/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1612 - val_loss: 0.2808
Epoch 60/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1452 - val_loss: 0.2540
Epoch 61/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1530 - val_loss: 0.2806
Epoch 62/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1285 - val_loss: 0.2720
Epoch 63/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1394 - val_loss: 0.2703
Epoch 64/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1305 - val_loss: 0.2747
Epoch 65/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1636 - val_loss: 0.2804
Epoch 66/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1229 - val_loss: 0.2798
Epoch 67/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1599 - val_loss: 0.2699
Epoch 68/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1312 - val_loss: 0.2805
Epoch 69/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1289 - val_loss: 0.2807
Epoch 70/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1260 - val_loss: 0.2778
Epoch 71/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1308 - val_loss: 0.2755
Epoch 72/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1516 - val_loss: 0.2843
Epoch 73/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1228 - val_loss: 0.2734
Epoch 74/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1388 - val_loss: 0.2892
Epoch 75/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1311 - val_loss: 0.2696
Epoch 76/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1266 - val_loss: 0.3007
Epoch 77/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1304 - val_loss: 0.3022
Epoch 78/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1236 - val_loss: 0.3183
Epoch 79/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1344 - val_loss: 0.3160
Epoch 80/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1397 - val_loss: 0.3159
Epoch 81/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1406 - val_loss: 0.3188
Epoch 82/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1391 - val_loss: 0.2909
Epoch 83/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1657 - val_loss: 0.3055
Epoch 84/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1431 - val_loss: 0.2749
Epoch 85/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1214 - val_loss: 0.2924
Epoch 86/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1405 - val_loss: 0.3231
Epoch 87/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1707 - val_loss: 0.2653
Epoch 88/100
17/17 [==============================] - ETA: 0s - loss: 0.129 - 0s 3ms/step - loss: 0.1154 - val_los
s: 0.2674
Epoch 89/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1291 - val_loss: 0.2781
Epoch 90/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1205 - val_loss: 0.3164
Epoch 91/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1252 - val_loss: 0.3060
Epoch 92/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1201 - val_loss: 0.3135
Epoch 93/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1300 - val_loss: 0.3077
Epoch 94/100
17/17 [==============================] - 0s 4ms/step - loss: 0.1269 - val_loss: 0.2885
Epoch 95/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1288 - val_loss: 0.3134
Epoch 96/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1175 - val_loss: 0.2968
Epoch 97/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1345 - val_loss: 0.3277
Epoch 98/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1094 - val_loss: 0.3033
Epoch 99/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1178 - val_loss: 0.3129
Epoch 100/100
17/17 [==============================] - 0s 3ms/step - loss: 0.1123 - val_loss: 0.2934
```

In [20]:
```python
result = ANN_model.evaluate(X_test, y_test)
accuracy_ANN = 1 - result
print("Accuracy : {}".format(accuracy_ANN))
```

```
3/3 [==============================] - 0s 1ms/step - loss: 0.1936
Accuracy : 0.8064108341932297
```

In [21]:
```python
epochs_hist.history.keys()
```

Out[21]: `dict_keys(['loss', 'val_loss'])`

In [22]:
```python
plt.plot(epochs_hist.history['loss'])
plt.title('Model Loss Progress During Training')
plt.xlabel('Epoch')
plt.ylabel('Training Loss')
plt.legend(['Training Loss'])
```

Out[22]: `<matplotlib.legend.Legend at 0x2dac55219c8>`

References

https://www.kaggle.com/mohamedchahrazed/graduate-admissions

https://www.coursera.org/projects/machine-learning-university-admission