UNDERSTAND THE PROBLEM STATEMENT

- In this project, we will build a regression model to predict the chance of admission into a particular university based on the student's profile.

- INPUTS (FEATURES):
  - GRE Scores (out of 340)
  - TOEFL Scores (out of 120)
  - University Rating (out of 5)
  - Statement of Purpose (SOP)
  - Letter of Recommendation (LOR) Strength (out of 5)
  - Undergraduate GPA (out of 10)
  - Research Experience (either 0 or 1)

- OUTPUTS:
  - Chance of admission (ranging from 0 to 1)

- Data Source: https://www.kaggle.com/mohansacharya/graduate-admissions
- Photo Credit: https://www.pexels.com/photo/accomplishment-ceremony-education-graduation-267885/

IMPORT LIBRARIES AND DATASET

In [3]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from jupyterthemes import jtplot
jtplot.style(theme='monokai', context='notebook', ticks=True, grid=False)
```

```
C:\Users\Administrator\anaconda31\lib\importlib\_bootstrap.py:219: RuntimeWarning: numpy.
ufunc size changed, may indicate binary incompatibility. Expected 192 from C header, got
216 from PyObject
  return f(*args, **kwds)
C:\Users\Administrator\anaconda31\lib\importlib\_bootstrap.py:219: RuntimeWarning: numpy.
ufunc size changed, may indicate binary incompatibility. Expected 192 from C header, got
216 from PyObject
  return f(*args, **kwds)
```

In [4]:

```
# read the csv file
admission_df = pd.read_csv('Admission_Predict_raw.csv')
```

In [5]:

```
admission_df.head()
```

Out[5]:

	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	1	337	118	4	4.5	4.5	9.65	1	0.92
1	2	324	107	4	4.0	4.5	8.87	1	0.76

2	Serial No.	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
3	4	322	110	3	3.5	2.5	8.67	1	0.80
4	5	314	103	2	2.0	3.0	8.21	0	0.65

In [6]:

```
# Let's drop the serial no.
admission_df.drop('Serial No.', axis = 1, inplace = True)
admission_df
```

Out[6]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
0	337	118	4	4.5	4.5	9.65	1	0.92
1	324	107	4	4.0	4.5	8.87	1	0.76
2	316	104	3	3.0	3.5	8.00	1	0.72
3	322	110	3	3.5	2.5	8.67	1	0.80
4	314	103	2	2.0	3.0	8.21	0	0.65
...	...	...	...	...	...	...	...	...
495	332	108	5	4.5	4.0	9.02	1	0.87
496	337	117	5	5.0	5.0	9.87	1	0.96
497	330	120	5	4.5	5.0	9.56	1	0.93
498	312	103	4	4.0	5.0	8.43	0	0.73
499	327	113	4	4.5	4.5	9.04	0	0.84

500 rows × 8 columns

PERFORM EXPLORATORY DATA ANALYSIS

In [7]:

```
# checking the null values
admission_df.isnull().sum()
```

Out[7]:

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

In [8]:

```
# Check the dataframe information
admission_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   GRE Score          500 non-null    int64
 1   TOEFL Score        500 non-null    int64
 2   University Rating  500 non-null    int64
 3   SOP                500 non-null    float64
 4   LOR                500 non-null    float64
```

```
 5   CGPA               500 non-null    float64
 6   Research           500 non-null    int64
 7   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(4)
memory usage: 31.4 KB
```

In [9]:

```
# Statistical summary of the dataframe
admission_df.describe()
```

Out[9]:

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research	Chance of Admit
count	500.000000	500.000000	500.000000	500.000000	500.00000	500.000000	500.000000	500.00000
mean	316.472000	107.192000	3.114000	3.374000	3.48400	8.576440	0.560000	0.72174
std	11.295148	6.081868	1.143512	0.991004	0.92545	0.604813	0.496884	0.14114
min	290.000000	92.000000	1.000000	1.000000	1.00000	6.800000	0.000000	0.34000
25%	308.000000	103.000000	2.000000	2.500000	3.00000	8.127500	0.000000	0.63000
50%	317.000000	107.000000	3.000000	3.500000	3.50000	8.560000	1.000000	0.72000
75%	325.000000	112.000000	4.000000	4.000000	4.00000	9.040000	1.000000	0.82000
max	340.000000	120.000000	5.000000	5.000000	5.00000	9.920000	1.000000	0.97000

In [10]:

```
# Grouping by University ranking
df_university = admission_df.groupby(by = 'University Rating').mean()
df_university
```

Out[10]:

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

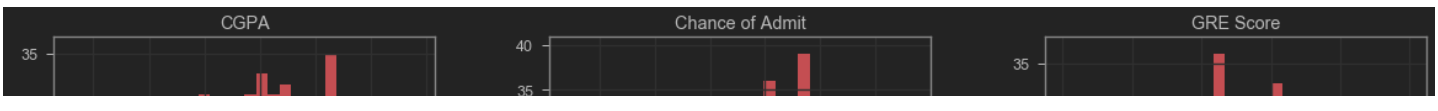
PERFORM DATA VISUALIZATION

In [11]:

```
admission_df.hist(bins = 30, figsize = (20, 20), color = 'r')
```

Out[11]:

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x00000291AF9BF2C8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000291B0107E48>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000291B013EB08>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000291B0177A88>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000291B01B0BC8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000291B01E9CC8>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x00000291B021FDC8>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000291B0257F08>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x00000291B0263A88>]],
      dtype=object)
```

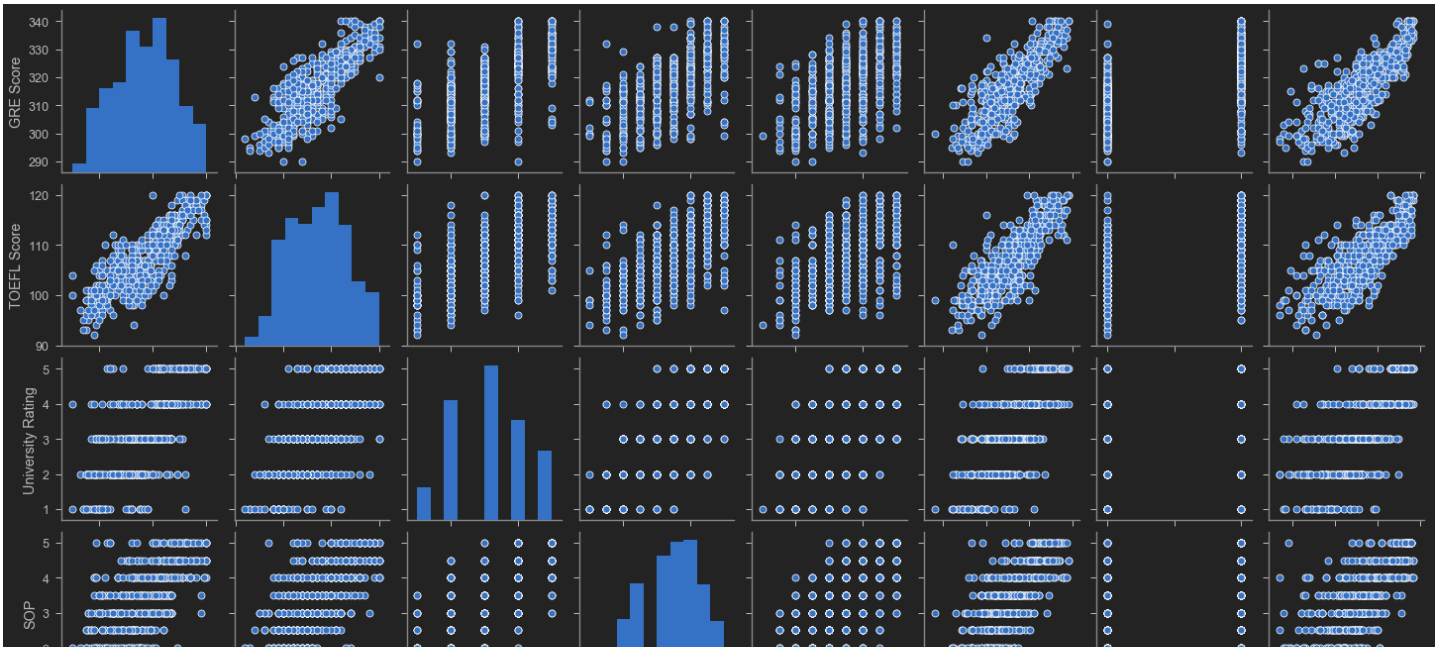


In [12]:

```
sns.pairplot(admission_df)
```

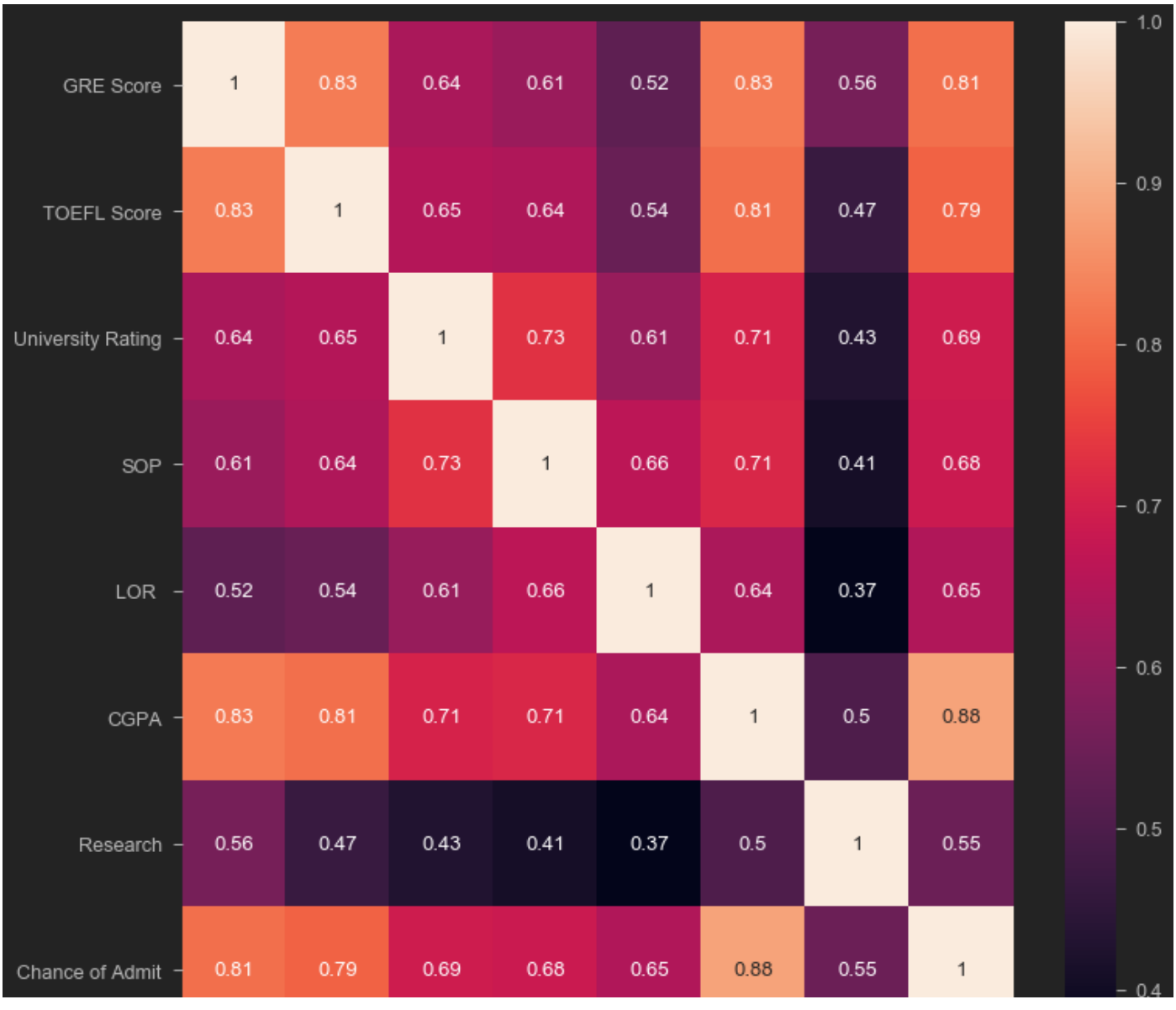
Out[12]:

<seaborn.axisgrid.PairGrid at 0x291b0e67c88>



In [13]:

```
corr_matrix = admission_df.corr()
plt.figure(figsize = (12, 12))
sns.heatmap(corr_matrix, annot = True)
plt.show()
```



CREATE TRAINING AND TESTING DATASET

In [43]:

```
admission_df.columns
```

Out[43]:

```
Index(['GRE Score', 'TOEFL Score', 'University Rating', 'SOP', 'LOR ', 'CGPA',
       'Research', 'Chance of Admit'],
      dtype='object')
```

In [44]:

```
X = admission_df.drop(columns = ['Chance of Admit']) #input
```

In [45]:

```
y = admission_df['Chance of Admit'] #outpu
```

In [46]:

```
X.shape
```

Out[46]:

```
(500, 7)
```

In [47]:

```
y.shape
```

Out[47]:

```
(500,)
```

In [48]:

```
y
```

Out[48]:

```
0      0.92
1      0.76
2      0.72
3      0.80
4      0.65
       ...
495    0.87
496    0.96
497    0.93
498    0.73
499    0.84
Name: Chance of Admit, Length: 500, dtype: float64
```

In [49]:

```
X = np.array(X)
y = np.array(y)
```

In [50]:

```
y = y.reshape(-1, 1)
y.shape
```

Out[50]:

```
(500, 1)
```

In [51]:

```
# scaling the data before training the model
from sklearn.preprocessing import StandardScaler, MinMaxScaler
scaler_x = StandardScaler()
X = scaler_x.fit_transform(X)
```

In [52]:

```
scaler_y = StandardScaler()
y = scaler_y.fit_transform(y)
```

In [53]:

```
# spliting the data in to test and train sets
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)
```

TRAIN AND EVALUATE A LINEAR REGRESSION MODEL

- In this project, we will build a regression model to predict the chance of admission into a particular university based on the student's profile.
- INPUTS (FEATURES):
  - GRE Scores (out of 340)
  - TOEFL Scores (out of 120)
  - University Rating (out of 5)
  - Statement of Purpose (SOP)
  - Letter of Recommendation (LOR) Strength (out of 5)
  - Undergraduate GPA (out of 10)
  - Research Experience (either 0 or 1)
- OUTPUTS:
  - Chance of admission (ranging from 0 to 1)

- Data Source: https://www.kaggle.com/mohansacharya/graduate-admissions
- Photo Credit: https://www.pexels.com/photo/accomplishment-ceremony-education-graduation-267885/

- In this project, we will build a regression model to predict the chance of admission into a particular university based on the student's profile.
- INPUTS (FEATURES):
  - GRE Scores (out of 340)
  - TOEFL Scores (out of 120)
  - University Rating (out of 5)
  - Statement of Purpose (SOP)
  - Letter of Recommendation (LOR) Strength (out of 5)
  - Undergraduate GPA (out of 10)

- Research Experience (either 0 or 1)

- OUTPUTS:
  - Chance of admission (ranging from 0 to 1)

- Data Source: https://www.kaggle.com/mohansacharya/graduate-admissions
- Photo Credit: https://www.pexels.com/photo/accomplishment-ceremony-education-graduation-267885/

- In this project, we will build a regression model to predict the chance of admission into a particular university based on the student's profile.

- INPUTS (FEATURES):
  - GRE Scores (out of 340)
  - TOEFL Scores (out of 120)
  - University Rating (out of 5)
  - Statement of Purpose (SOP)
  - Letter of Recommendation (LOR) Strength (out of 5)
  - Undergraduate GPA (out of 10)
  - Research Experience (either 0 or 1)

- OUTPUTS:
  - Chance of admission (ranging from 0 to 1)

- Data Source: https://www.kaggle.com/mohansacharya/graduate-admissions
- Photo Credit: https://www.pexels.com/photo/accomplishment-ceremony-education-graduation-267885/

In [54]:

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, accuracy_score
```

In [55]:

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

Out[55]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

In [56]:

```
accuracy_LinearRegression = LinearRegression_model.score(X_test, y_test)
accuracy_LinearRegression
```

Out[56]:

```
0.8452550081492437
```

TRAIN AND EVALUATE AN ARTIFICIAL NEURAL NETWORK

- In this project, we will build a regression model to predict the chance of

admission into a particular university based on the student's profile.

- **INPUTS (FEATURES):**
  - GRE Scores (out of 340)
  - TOEFL Scores (out of 120)
  - University Rating (out of 5)
  - Statement of Purpose (SOP)
  - Letter of Recommendation (LOR) Strength (out of 5)
  - Undergraduate GPA (out of 10)
  - Research Experience (either 0 or 1)
- **OUTPUTS:**
  - Chance of admission (ranging from 0 to 1)

- Data Source: https://www.kaggle.com/mohansacharya/graduate-admissions
- Photo Credit: https://www.pexels.com/photo/accomplishment-ceremony-education-graduation-267885/

In [57]:

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.layers import Dense, Activation, Dropout
from tensorflow.keras.optimizers import Adam
```

In [58]:

```
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))
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_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_5 (Dense)              (None, 50)                400
_________________________________________________________________
activation_4 (Activation)    (None, 50)                0
_________________________________________________________________
dense_6 (Dense)              (None, 150)               7650
_________________________________________________________________
activation_5 (Activation)    (None, 150)               0
_________________________________________________________________
dropout_2 (Dropout)          (None, 150)               0
_________________________________________________________________
dense_7 (Dense)              (None, 150)               22650
_________________________________________________________________
activation_6 (Activation)    (None, 150)               0
_________________________________________________________________
dropout_3 (Dropout)          (None, 150)               0
_________________________________________________________________
dense_8 (Dense)              (None, 50)                7550
_________________________________________________________________
activation_7 (Activation)    (None, 50)                0
_________________________________________________________________
dense_9 (Dense)              (None, 1)                 51
```

```
=================================================================
Total params: 38,301
Trainable params: 38,301
Non-trainable params: 0
_________________________________________________________________
```

In [59]:

```
ANN_model.compile(optimizer='Adam', loss='mean_squared_error')
```

In [60]:

```
epochs_hist = ANN_model.fit(X_train, y_train, epochs = 100, batch_size = 20, validation_
split = 0.2)
```

```
Train on 340 samples, validate on 85 samples
Epoch 1/100
340/340 [==============================] - 1s 3ms/sample - loss: 0.6247 - val_loss: 0.274
9
Epoch 2/100
340/340 [==============================] - 0s 324us/sample - loss: 0.4103 - val_loss: 0.2
645
Epoch 3/100
340/340 [==============================] - 0s 285us/sample - loss: 0.3322 - val_loss: 0.2
559
Epoch 4/100
340/340 [==============================] - 0s 303us/sample - loss: 0.2987 - val_loss: 0.2
428
Epoch 5/100
340/340 [==============================] - 0s 256us/sample - loss: 0.2798 - val_loss: 0.2
429
Epoch 6/100
340/340 [==============================] - 0s 282us/sample - loss: 0.2841 - val_loss: 0.2
520
Epoch 7/100
340/340 [==============================] - 0s 306us/sample - loss: 0.2676 - val_loss: 0.2
450
Epoch 8/100
340/340 [==============================] - 0s 285us/sample - loss: 0.2678 - val_loss: 0.2
584
Epoch 9/100
340/340 [==============================] - ETA: 0s - loss: 0.285 - 0s 282us/sample - loss
: 0.2769 - val_loss: 0.2353
Epoch 10/100
340/340 [==============================] - 0s 229us/sample - loss: 0.2522 - val_loss: 0.2
339
Epoch 11/100
340/340 [==============================] - 0s 306us/sample - loss: 0.2522 - val_loss: 0.2
808
Epoch 12/100
340/340 [==============================] - 0s 279us/sample - loss: 0.2450 - val_loss: 0.2
769
Epoch 13/100
340/340 [==============================] - 0s 265us/sample - loss: 0.2670 - val_loss: 0.2
375
Epoch 14/100
340/340 [==============================] - 0s 309us/sample - loss: 0.2216 - val_loss: 0.2
873
Epoch 15/100
340/340 [==============================] - 0s 227us/sample - loss: 0.2207 - val_loss: 0.2
408
Epoch 16/100
340/340 [==============================] - 0s 306us/sample - loss: 0.1965 - val_loss: 0.2
578
Epoch 17/100
340/340 [==============================] - 0s 309us/sample - loss: 0.1942 - val_loss: 0.2
382
Epoch 18/100
340/340 [==============================] - 0s 321us/sample - loss: 0.2180 - val_loss: 0.2
702
Epoch 19/100
340/340 [==============================] - 0s 268us/sample - loss: 0.2094 - val_loss: 0.2
```

766
Epoch 20/100
340/340 [==============================] - 0s 224us/sample - loss: 0.2173 - val_loss: 0.2
592
Epoch 21/100
340/340 [==============================] - 0s 241us/sample - loss: 0.2121 - val_loss: 0.2
511
Epoch 22/100
340/340 [==============================] - 0s 324us/sample - loss: 0.2306 - val_loss: 0.2
384
Epoch 23/100
340/340 [==============================] - 0s 294us/sample - loss: 0.1907 - val_loss: 0.2
879
Epoch 24/100
340/340 [==============================] - 0s 315us/sample - loss: 0.1960 - val_loss: 0.2
330
Epoch 25/100
340/340 [==============================] - 0s 321us/sample - loss: 0.2315 - val_loss: 0.2
422
Epoch 26/100
340/340 [==============================] - 0s 330us/sample - loss: 0.2217 - val_loss: 0.2
727
Epoch 27/100
340/340 [==============================] - 0s 297us/sample - loss: 0.1935 - val_loss: 0.2
829
Epoch 28/100
340/340 [==============================] - 0s 347us/sample - loss: 0.2101 - val_loss: 0.2
417
Epoch 29/100
340/340 [==============================] - 0s 318us/sample - loss: 0.1994 - val_loss: 0.2
434
Epoch 30/100
340/340 [==============================] - 0s 309us/sample - loss: 0.1838 - val_loss: 0.2
503
Epoch 31/100
340/340 [==============================] - 0s 309us/sample - loss: 0.2040 - val_loss: 0.2
420
Epoch 32/100
340/340 [==============================] - 0s 318us/sample - loss: 0.1734 - val_loss: 0.2
595
Epoch 33/100
340/340 [==============================] - 0s 327us/sample - loss: 0.2098 - val_loss: 0.2
657
Epoch 34/100
340/340 [==============================] - 0s 335us/sample - loss: 0.1954 - val_loss: 0.2
412
Epoch 35/100
340/340 [==============================] - 0s 244us/sample - loss: 0.1870 - val_loss: 0.2
774
Epoch 36/100
340/340 [==============================] - 0s 276us/sample - loss: 0.1979 - val_loss: 0.2
667
Epoch 37/100
340/340 [==============================] - 0s 371us/sample - loss: 0.1752 - val_loss: 0.2
452
Epoch 38/100
340/340 [==============================] - 0s 324us/sample - loss: 0.1757 - val_loss: 0.2
447
Epoch 39/100
340/340 [==============================] - 0s 453us/sample - loss: 0.1714 - val_loss: 0.2
535
Epoch 40/100
340/340 [==============================] - 0s 350us/sample - loss: 0.1866 - val_loss: 0.2
432
Epoch 41/100
340/340 [==============================] - 0s 224us/sample - loss: 0.1929 - val_loss: 0.2
444
Epoch 42/100
340/340 [==============================] - 0s 232us/sample - loss: 0.1774 - val_loss: 0.2
533
Epoch 43/100
340/340 [==============================] - 0s 250us/sample - loss: 0.1722 - val_loss: 0.2

660
Epoch 44/100
340/340 [==============================] - 0s 262us/sample - loss: 0.1886 - val_loss: 0.2
394
Epoch 45/100
340/340 [==============================] - 0s 271us/sample - loss: 0.1695 - val_loss: 0.2
418
Epoch 46/100
340/340 [==============================] - 0s 250us/sample - loss: 0.1644 - val_loss: 0.2
618
Epoch 47/100
340/340 [==============================] - 0s 212us/sample - loss: 0.1654 - val_loss: 0.2
714
Epoch 48/100
340/340 [==============================] - 0s 256us/sample - loss: 0.1604 - val_loss: 0.2
623
Epoch 49/100
340/340 [==============================] - 0s 256us/sample - loss: 0.1700 - val_loss: 0.2
484
Epoch 50/100
340/340 [==============================] - 0s 271us/sample - loss: 0.1636 - val_loss: 0.2
614
Epoch 51/100
340/340 [==============================] - 0s 232us/sample - loss: 0.1585 - val_loss: 0.2
577
Epoch 52/100
340/340 [==============================] - 0s 209us/sample - loss: 0.1586 - val_loss: 0.2
705
Epoch 53/100
340/340 [==============================] - 0s 227us/sample - loss: 0.1719 - val_loss: 0.2
792
Epoch 54/100
340/340 [==============================] - 0s 232us/sample - loss: 0.1745 - val_loss: 0.2
608
Epoch 55/100
340/340 [==============================] - 0s 232us/sample - loss: 0.1417 - val_loss: 0.2
380
Epoch 56/100
340/340 [==============================] - 0s 218us/sample - loss: 0.1411 - val_loss: 0.2
517
Epoch 57/100
340/340 [==============================] - 0s 224us/sample - loss: 0.1537 - val_loss: 0.2
657
Epoch 58/100
340/340 [==============================] - 0s 253us/sample - loss: 0.1604 - val_loss: 0.2
688
Epoch 59/100
340/340 [==============================] - 0s 241us/sample - loss: 0.1738 - val_loss: 0.2
452
Epoch 60/100
340/340 [==============================] - 0s 259us/sample - loss: 0.1590 - val_loss: 0.2
423
Epoch 61/100
340/340 [==============================] - 0s 226us/sample - loss: 0.1533 - val_loss: 0.2
438
Epoch 62/100
340/340 [==============================] - 0s 253us/sample - loss: 0.1541 - val_loss: 0.2
597
Epoch 63/100
340/340 [==============================] - 0s 288us/sample - loss: 0.1492 - val_loss: 0.2
490
Epoch 64/100
340/340 [==============================] - 0s 221us/sample - loss: 0.1395 - val_loss: 0.2
430
Epoch 65/100
340/340 [==============================] - 0s 227us/sample - loss: 0.1373 - val_loss: 0.2
412
Epoch 66/100
340/340 [==============================] - 0s 218us/sample - loss: 0.1429 - val_loss: 0.2
646
Epoch 67/100
340/340 [==============================] - 0s 244us/sample - loss: 0.1504 - val_loss: 0.2

719
Epoch 68/100
340/340 [==============================] - 0s 300us/sample - loss: 0.1374 - val_loss: 0.2
828
Epoch 69/100
340/340 [==============================] - 0s 291us/sample - loss: 0.1486 - val_loss: 0.2
633
Epoch 70/100
340/340 [==============================] - 0s 279us/sample - loss: 0.1534 - val_loss: 0.2
753
Epoch 71/100
340/340 [==============================] - 0s 341us/sample - loss: 0.1418 - val_loss: 0.2
740
Epoch 72/100
340/340 [==============================] - 0s 371us/sample - loss: 0.1324 - val_loss: 0.2
780
Epoch 73/100
340/340 [==============================] - 0s 259us/sample - loss: 0.1246 - val_loss: 0.2
770
Epoch 74/100
340/340 [==============================] - 0s 291us/sample - loss: 0.1322 - val_loss: 0.2
795
Epoch 75/100
340/340 [==============================] - 0s 244us/sample - loss: 0.1436 - val_loss: 0.2
589
Epoch 76/100
340/340 [==============================] - 0s 315us/sample - loss: 0.1334 - val_loss: 0.2
603
Epoch 77/100
340/340 [==============================] - 0s 377us/sample - loss: 0.1345 - val_loss: 0.2
541
Epoch 78/100
340/340 [==============================] - 0s 303us/sample - loss: 0.1543 - val_loss: 0.2
801
Epoch 79/100
340/340 [==============================] - 0s 191us/sample - loss: 0.1349 - val_loss: 0.2
501
Epoch 80/100
340/340 [==============================] - 0s 209us/sample - loss: 0.1361 - val_loss: 0.2
603
Epoch 81/100
340/340 [==============================] - 0s 200us/sample - loss: 0.1209 - val_loss: 0.2
782
Epoch 82/100
340/340 [==============================] - 0s 241us/sample - loss: 0.1445 - val_loss: 0.2
655
Epoch 83/100
340/340 [==============================] - 0s 194us/sample - loss: 0.1205 - val_loss: 0.2
637
Epoch 84/100
340/340 [==============================] - 0s 200us/sample - loss: 0.1387 - val_loss: 0.2
707
Epoch 85/100
340/340 [==============================] - 0s 250us/sample - loss: 0.1254 - val_loss: 0.2
627
Epoch 86/100
340/340 [==============================] - 0s 247us/sample - loss: 0.1285 - val_loss: 0.2
932
Epoch 87/100
340/340 [==============================] - 0s 244us/sample - loss: 0.1231 - val_loss: 0.2
672
Epoch 88/100
340/340 [==============================] - 0s 235us/sample - loss: 0.1362 - val_loss: 0.2
855
Epoch 89/100
340/340 [==============================] - 0s 268us/sample - loss: 0.1156 - val_loss: 0.2
699
Epoch 90/100
340/340 [==============================] - 0s 382us/sample - loss: 0.1103 - val_loss: 0.2
682
Epoch 91/100
340/340 [==============================] - 0s 206us/sample - loss: 0.1016 - val_loss: 0.2

```
897
Epoch 92/100
340/340 [==============================] - 0s 238us/sample - loss: 0.1244 - val_loss: 0.2
949
Epoch 93/100
340/340 [==============================] - 0s 200us/sample - loss: 0.1070 - val_loss: 0.2
748
Epoch 94/100
340/340 [==============================] - 0s 253us/sample - loss: 0.1370 - val_loss: 0.2
775
Epoch 95/100
340/340 [==============================] - 0s 203us/sample - loss: 0.1237 - val_loss: 0.2
846
Epoch 96/100
340/340 [==============================] - 0s 238us/sample - loss: 0.1344 - val_loss: 0.2
830
Epoch 97/100
340/340 [==============================] - 0s 191us/sample - loss: 0.1254 - val_loss: 0.2
779
Epoch 98/100
340/340 [==============================] - 0s 188us/sample - loss: 0.1162 - val_loss: 0.2
868
Epoch 99/100
340/340 [==============================] - 0s 191us/sample - loss: 0.1074 - val_loss: 0.2
890
Epoch 100/100
340/340 [==============================] - 0s 256us/sample - loss: 0.1249 - val_loss: 0.2
651
```

In [61]:

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

```
75/1 [==============================================================================
==============================================================================
==============================================================================
==============================================================================
==============================================================================
==============================================================================
==============================================================================
==============================================================================
==============================================================================
==============================================================================
==============================================================================
==============================================================================
==============================================================================
==============================================================================
==============================================================================
==============================================================================
==============================================================================
==============================================================================
==============================] - 0s 80us/sample - loss: 0.1122
Accuracy : 0.8420152533054351
```

In [62]:

```
epochs_hist.history.keys()
```

Out[62]:

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

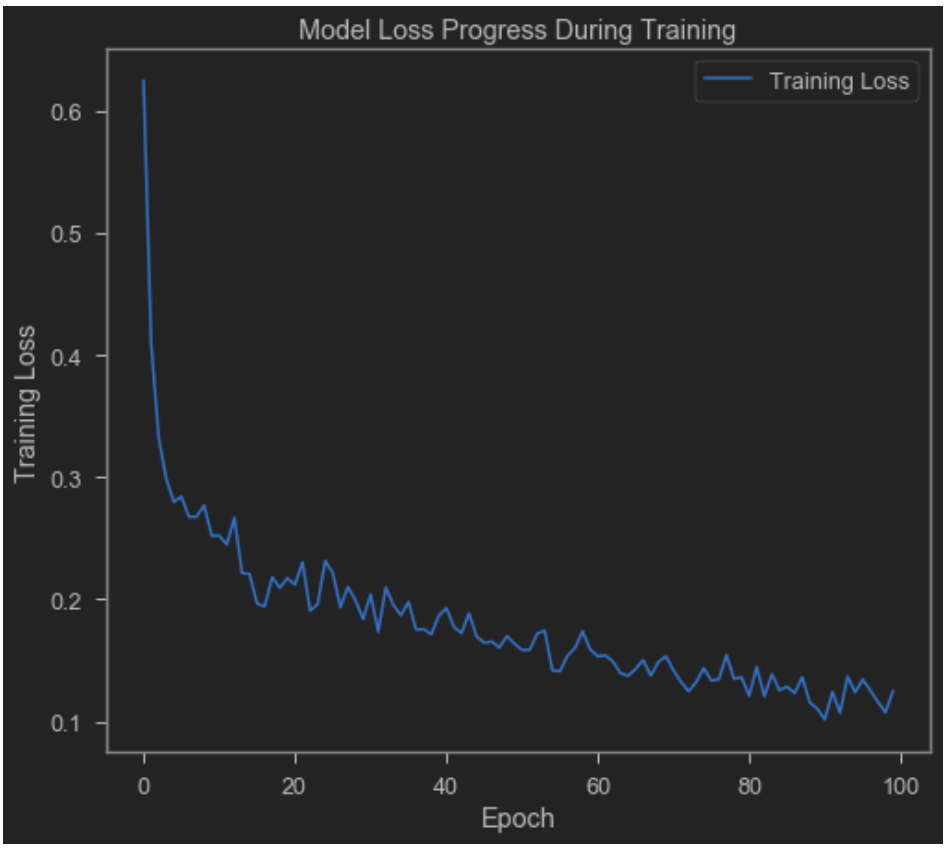
In [63]:

In [63]:

```
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[63]:

```
<matplotlib.legend.Legend at 0x291bcff84c8>
```



TRAIN AND EVALUATE A DECISION TREE AND RANDOM FOREST MODELS

In [64]:

```
# Decision tree builds regression or classification models in the form of a tree structur
e.
# Decision tree breaks down a dataset into smaller subsets while at the same time an asso
ciated decision tree is incrementally developed.
# The final result is a tree with decision nodes and leaf nodes.
# Great resource: https://www.saedsayad.com/decision_tree_reg.htm

from sklearn.tree import DecisionTreeRegressor
DecisionTree_model = DecisionTreeRegressor()
DecisionTree_model.fit(X_train, y_train)
```

Out[64]:

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

In [65]:

```
accuracy_DecisionTree = DecisionTree_model.score(X_test, y_test)
accuracy_DecisionTree
```

Out[65]:

```
0.5419644698615106
```

In [66]:

```
# Many decision Trees make up a random forest model which is an ensemble model.
# Predictions made by each decision tree are averaged to get the prediction of random for
est model.
# A random forest regressor fits a number of classifying decision trees on various sub-sa
mples of the dataset and uses averaging to improve the predictive accuracy and control ov
er-fitting.
```

In [67]:

```
from sklearn.ensemble import RandomForestRegressor
RandomForest_model = RandomForestRegressor(n_estimators = 100, max_depth = 10)
RandomForest_model.fit(X_train, y_train)
```

```
C:\Users\Administrator\anaconda31\lib\site-packages\ipykernel_launcher.py:3: DataConversi
onWarning: A column-vector y was passed when a 1d array was expected. Please change the s
hape of y to (n_samples,), for example using ravel().
  This is separate from the ipykernel package so we can avoid doing imports until
```

Out[67]:

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=10, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

In [68]:

```
accuracy_RandomForest = RandomForest_model.score(X_test, y_test)
accuracy_RandomForest
```

Out[68]:

```
0.7696051071976353
```

UNDERSTAND VARIOUS REGRESSION KPIs

- In this project, we will build a regression model to predict the chance of admission into a particular university based on the student's profile.
- INPUTS (FEATURES):
  - GRE Scores (out of 340)
  - TOEFL Scores (out of 120)
  - University Rating (out of 5)
  - Statement of Purpose (SOP)
  - Letter of Recommendation (LOR) Strength (out of 5)
  - Undergraduate GPA (out of 10)
  - Research Experience (either 0 or 1)
- OUTPUTS:
  - Chance of admission (ranging from 0 to 1)

- Data Source: https://www.kaggle.com/mohansacharya/graduate-admissions
- Photo Credit: https://www.pexels.com/photo/accomplishment-ceremony-education-graduation-267885/

- In this project, we will build a regression model to predict the chance of admission into a particular university based on the student's profile.

- INPUTS (FEATURES):
  - GRE Scores (out of 340)
  - TOEFL Scores (out of 120)
  - University Rating (out of 5)
  - Statement of Purpose (SOP)
  - Letter of Recommendation (LOR) Strength (out of 5)
  - Undergraduate GPA (out of 10)
  - Research Experience (either 0 or 1)

- OUTPUTS:
  - Chance of admission (ranging from 0 to 1)

- Data Source: https://www.kaggle.com/mohansacharya/graduate-admissions
- Photo Credit: https://www.pexels.com/photo/accomplishment-ceremony-education-graduation-267885/

- In this project, we will build a regression model to predict the chance of admission into a particular university based on the student's profile.

- INPUTS (FEATURES):
  - GRE Scores (out of 340)
  - TOEFL Scores (out of 120)
  - University Rating (out of 5)
  - Statement of Purpose (SOP)
  - Letter of Recommendation (LOR) Strength (out of 5)
  - Undergraduate GPA (out of 10)
  - Research Experience (either 0 or 1)

- OUTPUTS:
  - Chance of admission (ranging from 0 to 1)

- Data Source: https://www.kaggle.com/mohansacharya/graduate-admissions
- Photo Credit: https://www.pexels.com/photo/accomplishment-ceremony-education-graduation-267885/

- In this project, we will build a regression model to predict the chance of admission into a particular university based on the student's profile.

- INPUTS (FEATURES):
  - GRE Scores (out of 340)
  - TOEFL Scores (out of 120)
  - University Rating (out of 5)
  - Statement of Purpose (SOP)
  - Letter of Recommendation (LOR) Strength (out of 5)
  - Undergraduate GPA (out of 10)
  - Research Experience (either 0 or 1)

- OUTPUTS:
  - Chance of admission (ranging from 0 to 1)

- Data Source: https://www.kaggle.com/mohansacharya/graduate-admissions
- Photo Credit: https://www.pexels.com/photo/accomplishment-ceremony-education-graduation-267885/

- In this project, we will build a regression model to predict the chance of admission into a particular university based on the student's profile.
- INPUTS (FEATURES):
  - GRE Scores (out of 340)
  - TOEFL Scores (out of 120)
  - University Rating (out of 5)
  - Statement of Purpose (SOP)
  - Letter of Recommendation (LOR) Strength (out of 5)
  - Undergraduate GPA (out of 10)
  - Research Experience (either 0 or 1)
- OUTPUTS:
  - Chance of admission (ranging from 0 to 1)

- Data Source: https://www.kaggle.com/mohansacharya/graduate-admissions
- Photo Credit: https://www.pexels.com/photo/accomplishment-ceremony-education-graduation-267885/

- In this project, we will build a regression model to predict the chance of admission into a particular university based on the student's profile.
- INPUTS (FEATURES):
  - GRE Scores (out of 340)
  - TOEFL Scores (out of 120)
  - University Rating (out of 5)
  - Statement of Purpose (SOP)
  - Letter of Recommendation (LOR) Strength (out of 5)
  - Undergraduate GPA (out of 10)
  - Research Experience (either 0 or 1)
- OUTPUTS:
  - Chance of admission (ranging from 0 to 1)

- Data Source: https://www.kaggle.com/mohansacharya/graduate-admissions
- Photo Credit: https://www.pexels.com/photo/accomplishment-ceremony-education-graduation-267885/

- In this project, we will build a regression model to predict the chance of admission into a particular university based on the student's profile.

- INPUTS (FEATURES):
  - GRE Scores (out of 340)
  - TOEFL Scores (out of 120)
  - University Rating (out of 5)
  - Statement of Purpose (SOP)
  - Letter of Recommendation (LOR) Strength (out of 5)
  - Undergraduate GPA (out of 10)
  - Research Experience (either 0 or 1)
- OUTPUTS:
  - Chance of admission (ranging from 0 to 1)

- Data Source: https://www.kaggle.com/mohansacharya/graduate-admissions
- Photo Credit: https://www.pexels.com/photo/accomplishment-ceremony-education-graduation-267885/

- In this project, we will build a regression model to predict the chance of admission into a particular university based on the student's profile.
- INPUTS (FEATURES):
  - GRE Scores (out of 340)
  - TOEFL Scores (out of 120)
  - University Rating (out of 5)
  - Statement of Purpose (SOP)
  - Letter of Recommendation (LOR) Strength (out of 5)
  - Undergraduate GPA (out of 10)
  - Research Experience (either 0 or 1)
- OUTPUTS:
  - Chance of admission (ranging from 0 to 1)

- Data Source: https://www.kaggle.com/mohansacharya/graduate-admissions
- Photo Credit: https://www.pexels.com/photo/accomplishment-ceremony-education-graduation-267885/

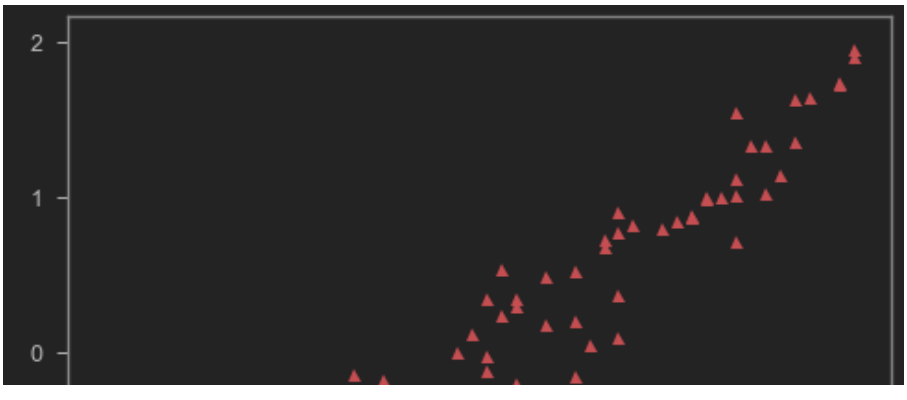
CALCULATE REGRESSION MODEL KPIs

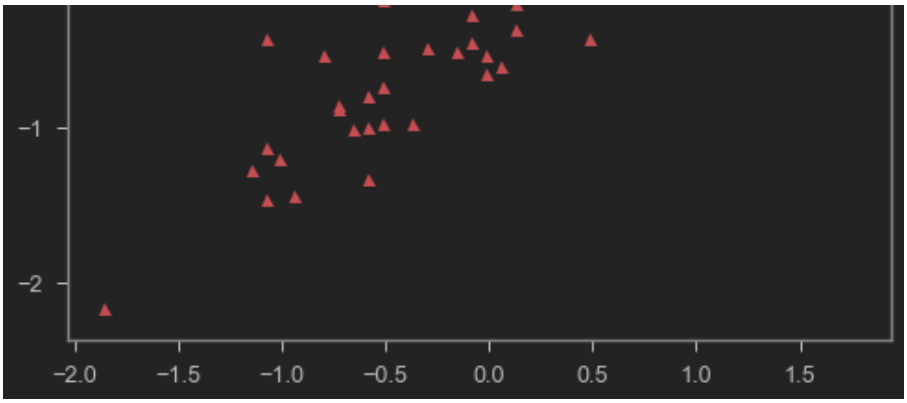
In [69]:

```
y_predict = LinearRegression_model.predict(X_test)
plt.plot(y_test, y_predict, '^', color = 'r')
```

Out[69]:

```
[<matplotlib.lines.Line2D at 0x291bd194748>]
```





In [70]:

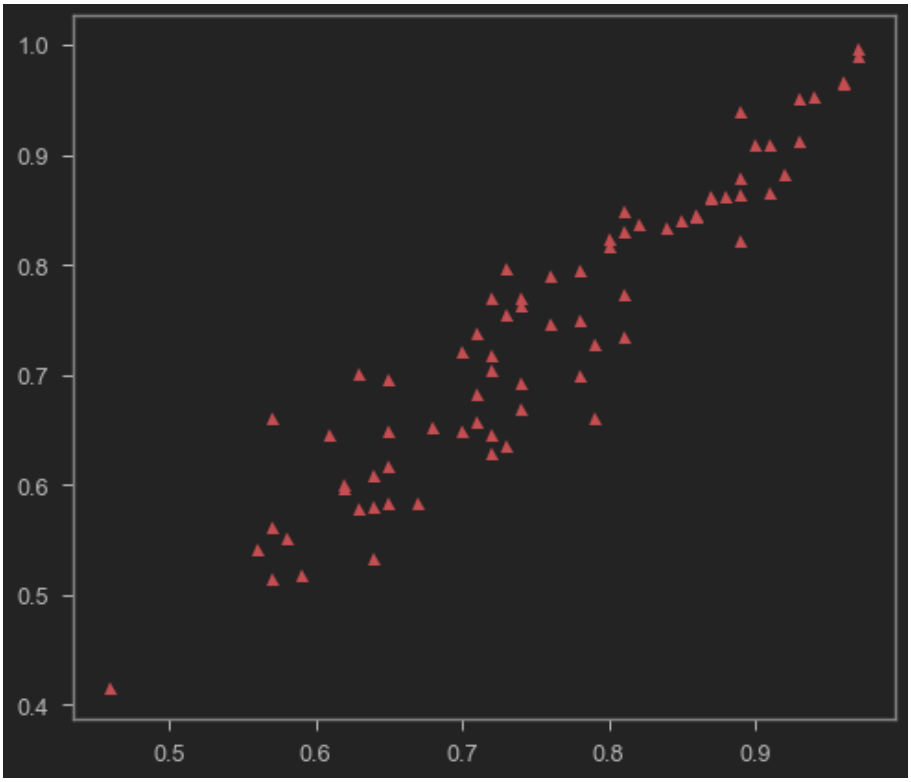
```
y_predict_orig = scaler_y.inverse_transform(y_predict)
y_test_orig = scaler_y.inverse_transform(y_test)
```

In [71]:

```
plt.plot(y_test_orig, y_predict_orig, '^', color = 'r')
```

Out[71]:

```
[<matplotlib.lines.Line2D at 0x291bd082908>]
```



In [72]:

```
k = X_test.shape[1]
n = len(X_test)
n
```

Out[72]:

```
75
```

In [73]:

```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
from math import sqrt

RMSE = float(format(np.sqrt(mean_squared_error(y_test_orig, y_predict_orig)),'.3f'))
MSE = mean_squared_error(y_test_orig, y_predict_orig)
MAE = mean_absolute_error(y_test_orig, y_predict_orig)
r2 = r2_score(y_test_orig, y_predict_orig)
```

```
adj_r2 = 1-(1-r2)*(n-1)/(n-k-1)

print('RMSE =',RMSE, '\nMSE =',MSE, '\nMAE =',MAE, '\nR2 =', r2, '\nAdjusted R2 =', adj_r
2)
```

```
RMSE = 0.046
MSE = 0.0021608563151818833
MAE = 0.0371989239418192
R2 = 0.8452550081492437
Adjusted R2 = 0.8290876209409557
```