```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

```
df=pd.read_csv('/content/Admission_Predict_Ver1.1.csv')
```

```
df.head()
```

	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	3	316	104	3	3.0	3.5	8.00	1	0.72
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

Next steps: Generate code with df View recommended plots

```
df.shape
```

```
(500, 9)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 500 entries, 0 to 499
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         500 non-null    int64  
 1   GRE Score          500 non-null    int64  
 2   TOEFL Score        500 non-null    int64  
 3   University Rating  500 non-null    int64  
 4   SOP                500 non-null    float64
 5   LOR                500 non-null    float64
 6   CGPA               500 non-null    float64
 7   Research           500 non-null    int64  
 8   Chance of Admit    500 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 35.3 KB
```

```
df.duplicated().sum()
```

```
0
```

```
df.drop(columns=['Serial No.'],inplace=True)
```

```
df.head()
```

	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

Next steps: Generate code with df View recommended plots

```
X = df.iloc[:,0:-1]
y = df.iloc[:,-1]
```

```
X
```

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

500 rows × 7 columns

Next steps: Generate code with X View recommended plots

```
y
```

```
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
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2,random_state=42)
```

```
X_train
```

	GRE Score	TOEFL Score	University Rating	SOP	LOR	CGPA	Research
249	321	111	3	3.5	4.0	8.83	1
433	316	111	4	4.0	5.0	8.54	0
19	303	102	3	3.5	3.0	8.50	0
322	314	107	2	2.5	4.0	8.27	0
332	308	106	3	3.5	2.5	8.21	1
...	...	...	...	...	...	...	...
106	329	111	4	4.5	4.5	9.18	1
270	306	105	2	2.5	3.0	8.22	1
348	302	99	1	2.0	2.0	7.25	0
435	309	105	2	2.5	4.0	7.68	0
102	314	106	2	4.0	3.5	8.25	0

400 rows × 7 columns

Next steps: Generate code with X_train View recommended plots

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
X_train_scaled
```

```
array([[0.62      , 0.67857143, 0.5       , ..., 0.71428571, 0.65064103,
        1.        ],
       [0.52      , 0.67857143, 0.75      , ..., 1.        , 0.55769231,
```

```
         0.        ],
        [0.26      , 0.35714286, 0.5       , ..., 0.42857143, 0.54487179,
         0.        ],
        ...,
        [0.24      , 0.25      , 0.        , ..., 0.14285714, 0.14423077,
         0.        ],
        [0.38      , 0.46428571, 0.25      , ..., 0.71428571, 0.28205128,
         0.        ],
        [0.48      , 0.5       , 0.25      , ..., 0.57142857, 0.46474359,
         0.        ]])
```

```
import tensorflow
from tensorflow import keras
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
```

```
model = Sequential()
model.add(Dense(7,activation='relu',input_dim=7))
model.add(Dense(7,activation='relu'))
model.add(Dense(1,activation='linear'))
```

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 7)                 56

 dense_1 (Dense)             (None, 7)                 56

 dense_2 (Dense)             (None, 1)                 8

=================================================================
Total params: 120 (480.00 Byte)
Trainable params: 120 (480.00 Byte)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

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

```
history = model.fit(X_train_scaled,y_train,epochs=150,validation_split=0.2)
```

```
Epoch 143/150
10/10 [==============================] - 0s 7ms/step - loss: 0.0042 - val_loss: 0.0030
Epoch 144/150
10/10 [==============================] - 0s 7ms/step - loss: 0.0042 - val_loss: 0.0031
Epoch 145/150
10/10 [==============================] - 0s 5ms/step - loss: 0.0042 - val_loss: 0.0030
Epoch 146/150
10/10 [==============================] - 0s 5ms/step - loss: 0.0042 - val_loss: 0.0032
Epoch 147/150
10/10 [==============================] - 0s 5ms/step - loss: 0.0042 - val_loss: 0.0030
Epoch 148/150
10/10 [==============================] - 0s 7ms/step - loss: 0.0042 - val_loss: 0.0031
Epoch 149/150
10/10 [==============================] - 0s 5ms/step - loss: 0.0041 - val_loss: 0.0030
Epoch 150/150
10/10 [==============================] - 0s 5ms/step - loss: 0.0042 - val_loss: 0.0030
```

```
y_pred = model.predict(X_test_scaled)
```

```
4/4 [==============================] - 0s 3ms/step
```

```
from sklearn.metrics import r2_score
r2_score(y_test,y_pred)
```

```
0.7819360247826792
```

```
import matplotlib.pyplot as plt
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
```

```
[<matplotlib.lines.Line2D at 0x79f986c7b3a0>]
```