Name: Navneet Yadav

Roll no. 2301560044

Program: MCA-I

Subject: AI-ML

Question 1

Rregression Modelling

GitHub: https://github.com/navYadav20/AI-ML (https://github.com/navYadav20/AI-ML)

In [50]:
```
import numpy as np
import pandas as pd
```

In [51]:
```
data = pd.read_csv("D:/ai-ml assignment/assignment 3/Real estate.csv")
data.head()
```

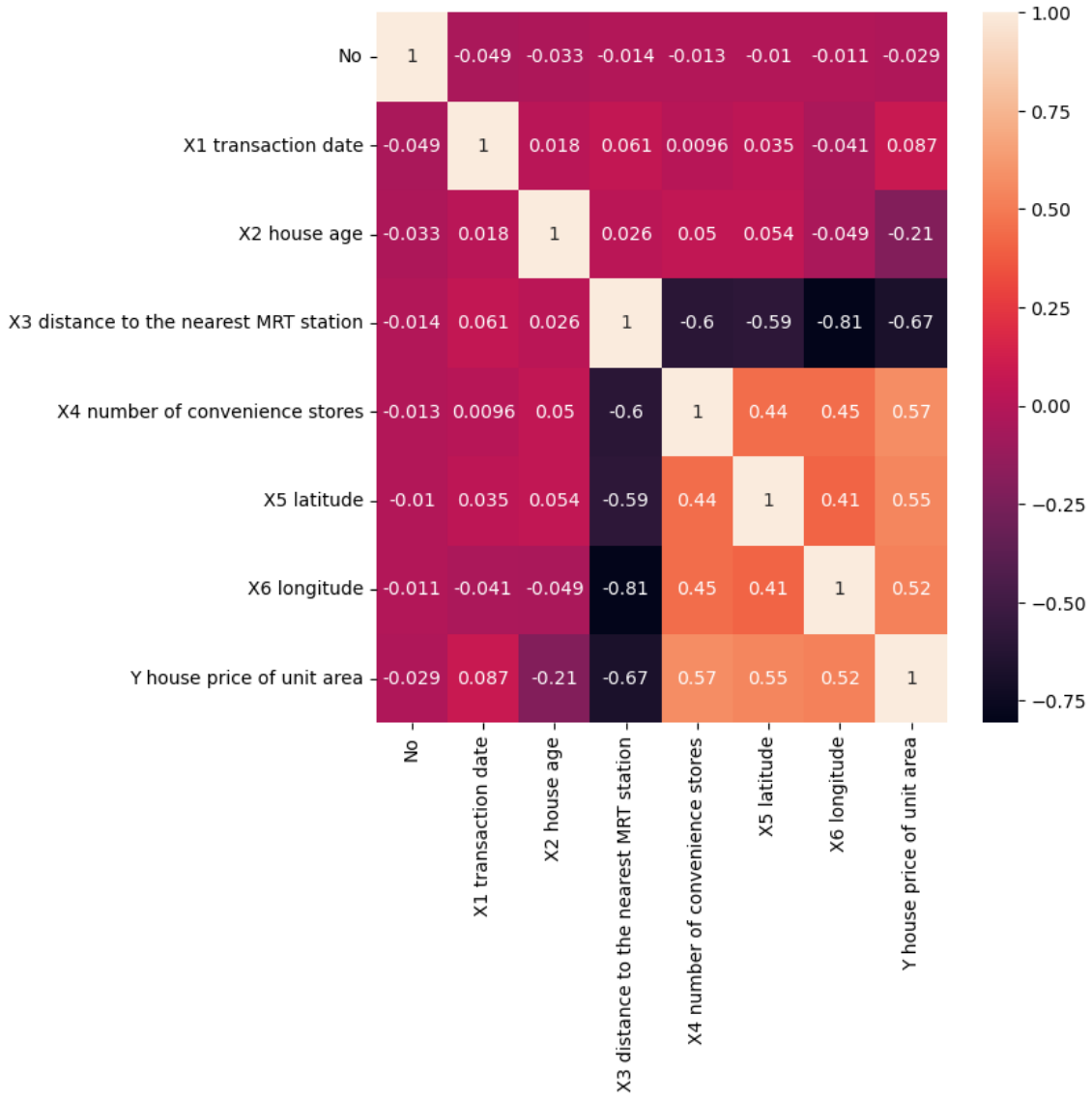
Out[51]:

	No	X1 transaction date	X2 house age	X3 distance to the nearest MRT station	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	1	2012.917	32.0	84.87882	10	24.98298	121.54024	37.9
1	2	2012.917	19.5	306.59470	9	24.98034	121.53951	42.2
2	3	2013.583	13.3	561.98450	5	24.98746	121.54391	47.3
3	4	2013.500	13.3	561.98450	5	24.98746	121.54391	54.8
4	5	2012.833	5.0	390.56840	5	24.97937	121.54245	43.1

In [52]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

plt.figure(figsize=(7,7))
sns.heatmap(data.corr(),annot=True)
```

Out[52]: <Axes: >



- Since the date feature is not appropriate to train with, i'll convert it to just years and remove features which gives negative correlations

In [53]:
```
unwanted = ['No','X2 house age','X3 distance to the nearest MRT station']
data = data.drop(unwanted,axis=1)
data.head()
```

Out[53]:

	X1 transaction date	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	2012.917	10	24.98298	121.54024	37.9
1	2012.917	9	24.98034	121.53951	42.2
2	2013.583	5	24.98746	121.54391	47.3
3	2013.500	5	24.98746	121.54391	54.8
4	2012.833	5	24.97937	121.54245	43.1

In [54]:
```
data['X1 transaction date'] = data['X1 transaction date'].astype(int)
data.head()
```

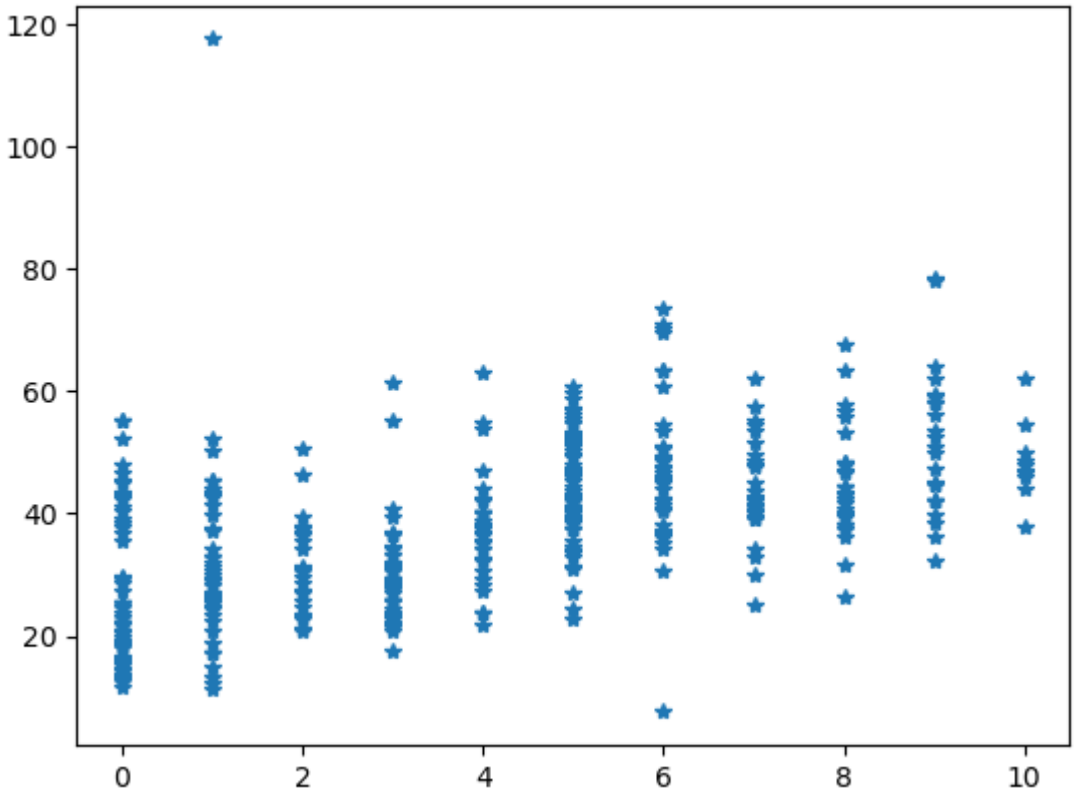
Out[54]:

	X1 transaction date	X4 number of convenience stores	X5 latitude	X6 longitude	Y house price of unit area
0	2012	10	24.98298	121.54024	37.9
1	2012	9	24.98034	121.53951	42.2
2	2013	5	24.98746	121.54391	47.3
3	2013	5	24.98746	121.54391	54.8
4	2012	5	24.97937	121.54245	43.1

Now the data looks clean, I'll plot some graphs for visualizing purposes

In [55]:
```
plt.plot(data['X4 number of convenience stores'],data['Y house price of uni
```

Out[55]:
```
[<matplotlib.lines.Line2D at 0x1bb21546f10>]
```

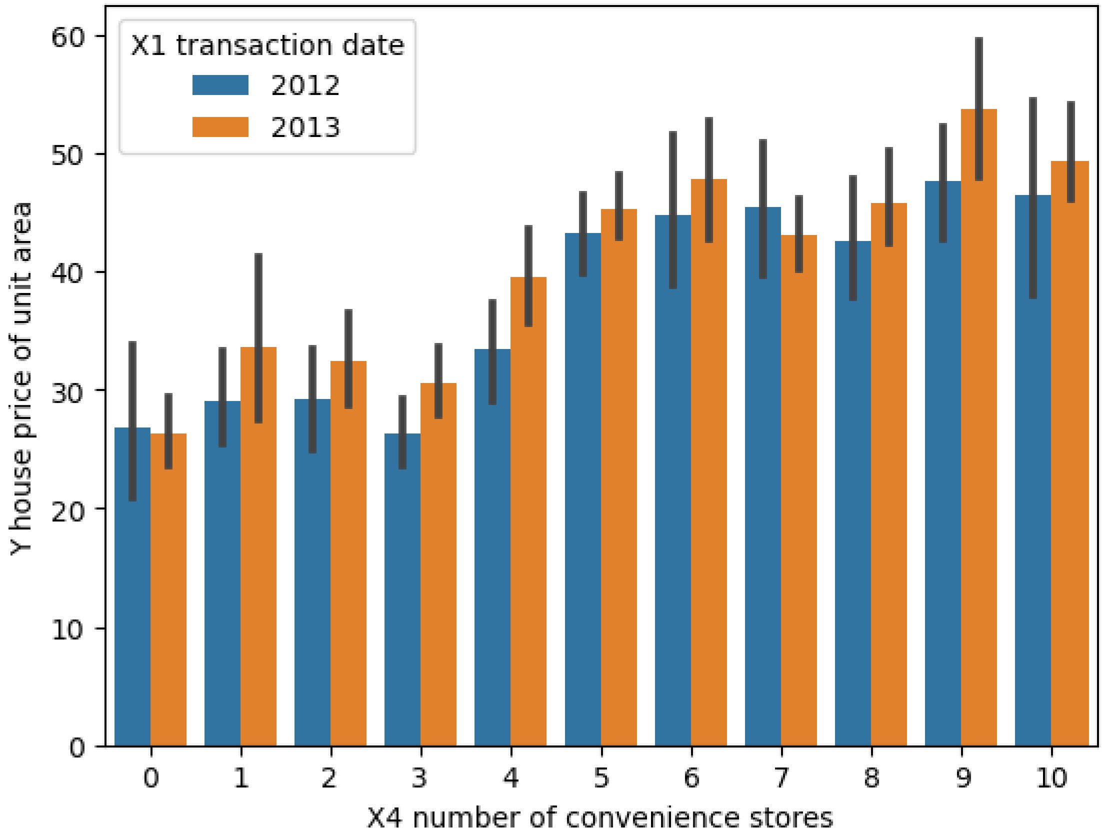


Splitting the data

In [9]:
```
sns.barplot(x=data['X4 number of convenience stores'],y=data['Y house price
```

```
C:\Users\msuse\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Futur
eWarning: is_categorical_dtype is deprecated and will be removed in a futu
re version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\msuse\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Futur
eWarning: is_categorical_dtype is deprecated and will be removed in a futu
re version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\msuse\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Futur
eWarning: is_categorical_dtype is deprecated and will be removed in a futu
re version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
C:\Users\msuse\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1498: Futur
eWarning: is_categorical_dtype is deprecated and will be removed in a futu
re version. Use isinstance(dtype, CategoricalDtype) instead
  if pd.api.types.is_categorical_dtype(vector):
```

Out[9]:
```
<Axes: xlabel='X4 number of convenience stores', ylabel='Y house price of
unit area'>
```



- we can understand that the date and number of convenient stores directly correlates with the price
- Now I split the data into train and test

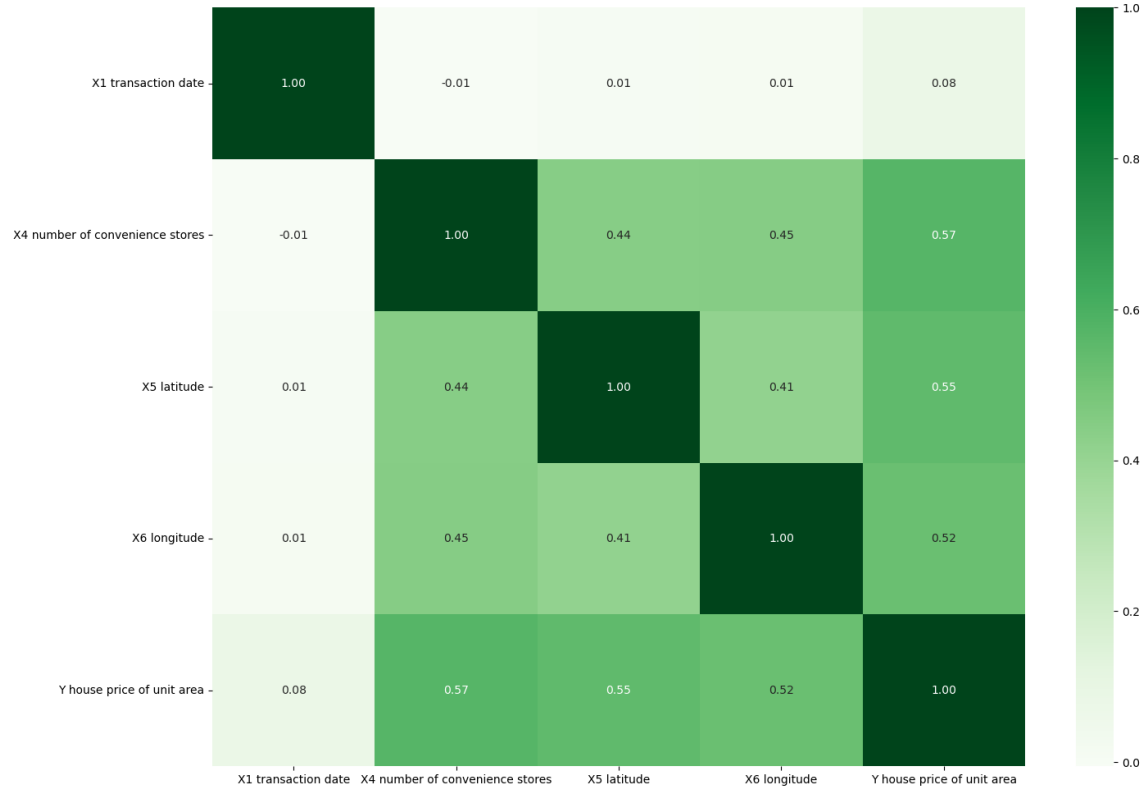
In [56]:
```
from sklearn.model_selection import train_test_split
X = data['X4 number of convenience stores'].values
y = data['Y house price of unit area'].values
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3)
print(X_train.shape,X_test.shape)
print(y_train.shape,y_test.shape)
```

```
(289,) (125,)
(289,) (125,)
```

Correlation matrix

In [102]:
```
correlation_matrix = data.corr()
```

In [103]:
```
plt.figure(figsize=(16,12))
sns.heatmap(correlation_matrix, annot=True,fmt=".2f", cmap="Greens")
plt.show()
```



Model Training

In [57]:
```
from sklearn.linear_model import LinearRegression
from sklearn import metrics

Lr = LinearRegression()
Lr.fit(X_train.reshape(-1,1),y_train)
```

Out[57]:
```
LinearRegression()
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

Prediction of model

In [58]:
```
y_pred = Lr.predict(X_test.reshape(-1,1))
print(y_pred.shape)
```

```
(125,)
```

Model Evaluation

Mean squared error as my metrics

In [59]:
```
from sklearn.metrics import mean_squared_error,accuracy_score, mean_absolut
print("MSE",mean_squared_error(y_test,y_pred)/100)
```

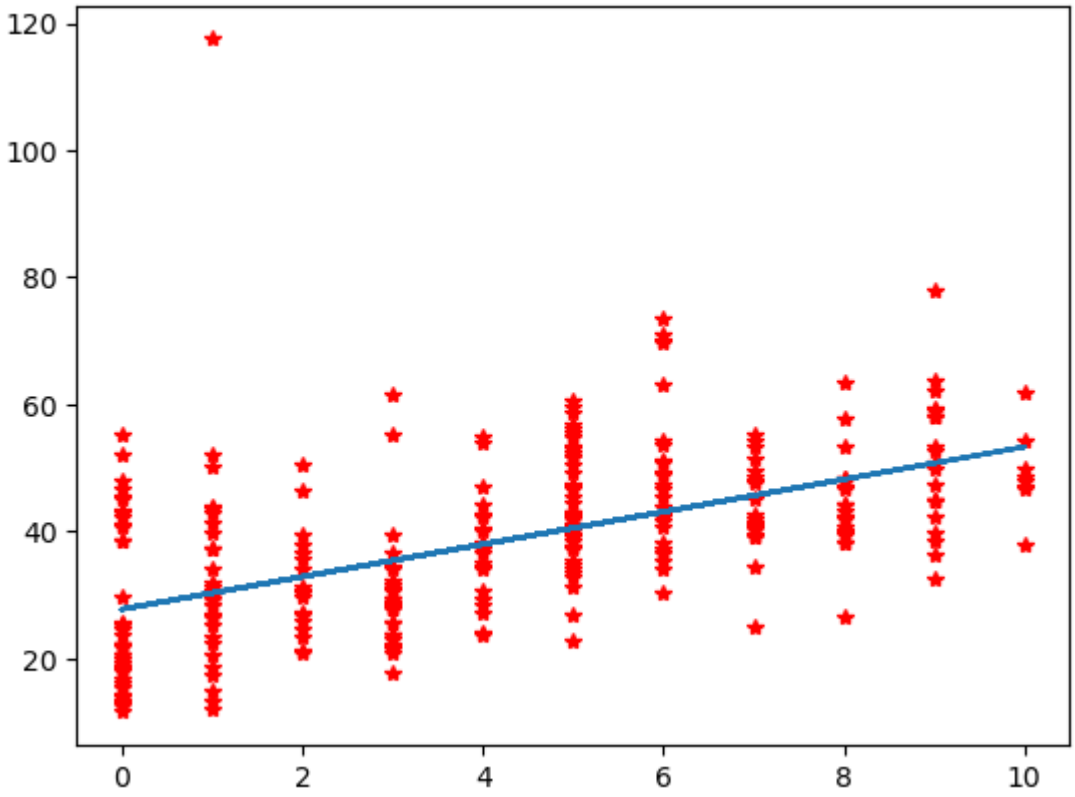
```
MSE 1.078751106619768
```

In [60]:
```
print(X_train.shape)
print(y_train.shape)
```

```
(289,)
(289,)
```

In [61]:
```
plt.plot(X_train,y_train,"r*")
plt.plot(X_test,y_pred)
```

Out[61]:
```
[<matplotlib.lines.Line2D at 0x1bb2168e750>]
```



In [62]:
```
print(X_test.shape)
print(y_test.shape)
```

```
(125,)
(125,)
```

In [63]:
```
import tensorflow as tf
```

- lets try with ANNs(Artificial Neural Networks)

In [64]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense

model = Sequential()

model.add(Dense(20,activation="relu",input_shape = (1,)))
model.add(Dense(20,activation="relu"))
model.add(Dense(1,activation="linear"))
```

Mean absolute error

In [65]:
```
from tensorflow.keras.optimizers import Adam
model.compile(loss="mean_absolute_error",optimizer = Adam(learning_rate=0.0
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_3 (Dense)             (None, 20)                40

 dense_4 (Dense)             (None, 20)                420

 dense_5 (Dense)             (None, 1)                 21

=================================================================
Total params: 481 (1.88 KB)
Trainable params: 481 (1.88 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

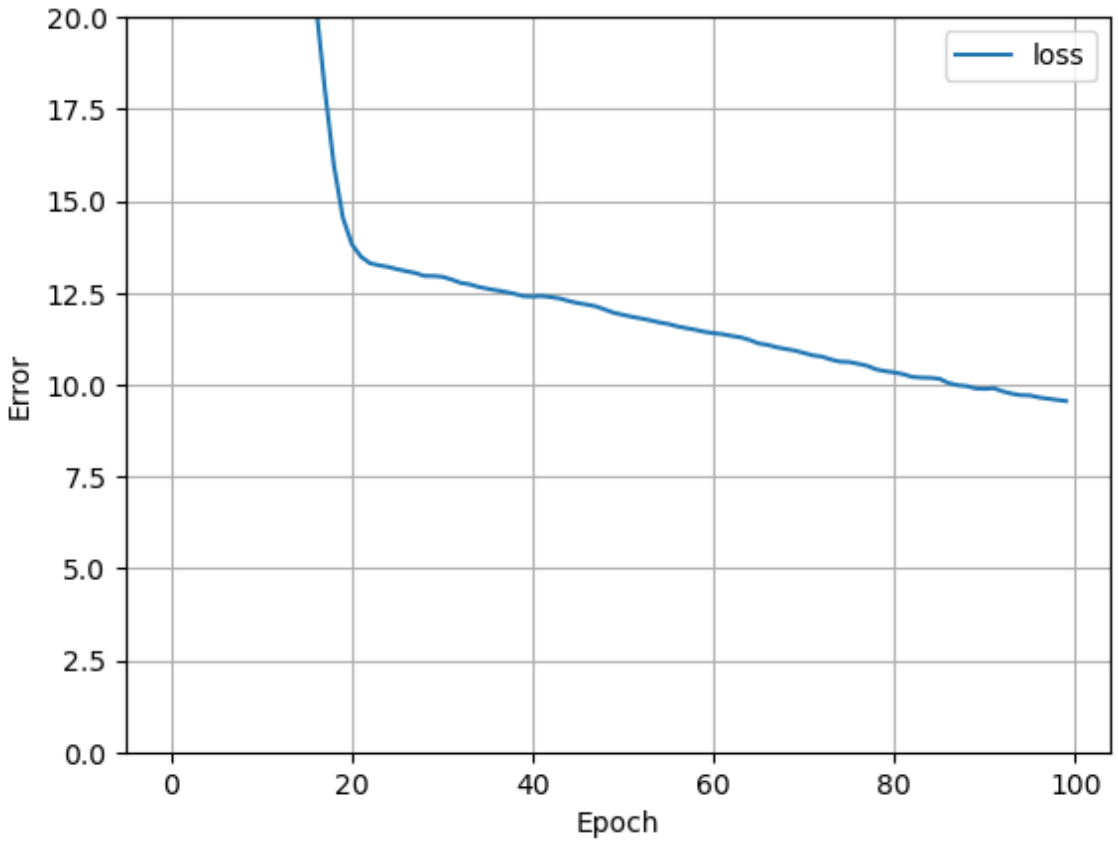
In [66]:
```
print("MAE",mean_absolute_error(y_test, y_pred))
```

```
MAE 8.343426611344011
```

In [67]:
```
history = model.fit(X_train,y_train,epochs=100,verbose=0)
```

In [68]:
```
def plot_loss(history):
    plt.plot(history.history['loss'], label='loss')
    plt.ylim([0, 20])
    plt.xlabel('Epoch')
    plt.ylabel('Error')
    plt.legend()
    plt.grid(True)
```

In [69]:
```
plot_loss(history)
```



We can see it is suffering from overfitting, let's try to make some changes

After tuning some hyper parameters such as the learning rate, number of layers and neurons we can conclude that the model has done a good job. Let's plot some insights

In [70]:
```
model.evaluate(X_test,y_test)
```

```
4/4 [==============================] - 0s 7ms/step - loss: 9.3343
```

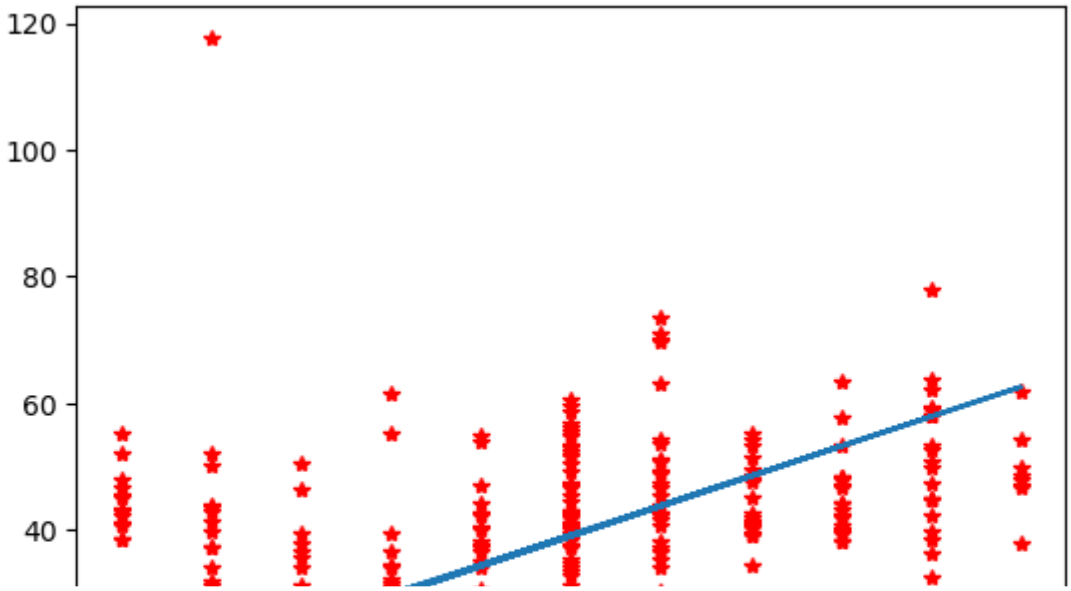
Out[70]:
```
9.334254264831543
```

In [71]:
```
import tensorflow as tf
x = tf.linspace(0.0,25.0,125)
y = model.predict(X_test)
y.reshape(125)
print(x.shape,y.shape)
```

```
4/4 [==============================] - 0s 2ms/step
(125,) (125, 1)
```

In [72]:
```
plt.plot(X_train,y_train,"r*")
plt.plot(X_test,y)
```

Out[72]:
```
[<matplotlib.lines.Line2D at 0x1bb21ce0ad0>]
```



Regression analysis metrics

In [86]:
```
MSE = mean_squared_error(y_test,y)/100
print("MSE",MSE)
```

```
MSE 1.4638761259304318
```

In [83]:
```
print("MAE",mean_absolute_error(y_test, y)/100)
```

```
MAE 0.09334254167175293
```

In [101]:
```
RMSE= np.sqrt(MSE)
print("Root Mean Squared Error",RMSE)
```

```
Root Mean Squared Error 1.2099074865172261
```

In [ ]:
```

```