```
# import the necessary libraries
from sklearn.datasets import load_boston
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
from statsmodels.stats.outliers_influence import variance_inflation_factor as vif
from sklearn.decomposition import PCA
from sklearn.preprocessing import RobustScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense

#load the dataset
data = load_boston()
#convert the dataset into a Pandas dataframe and add the target column named 'Price'
df = pd.DataFrame(data.data, columns=data.feature_names)
df['Price'] = data.target
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the functions in the public API at pandas.testing instead.
  import pandas.util.testing as tm
```

```
df.head()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
0	0.00632	18.0	2.31	0.0	0.538	6.575	65.2	4.0900	1.0	296.0	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0.0	0.469	6.421	78.9	4.9671	2.0	242.0	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0.0	0.469	7.185	61.1	4.9671	2.0	242.0	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0.0	0.458	6.998	45.8	6.0622	3.0	222.0	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0.0	0.458	7.147	54.2	6.0622	3.0	222.0	18.7	396.90	5.33	36.2

```
df.shape
```

```
(506, 14)
```

```
df.describe()
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	Price
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574901	3.795043	9.549407	408.237154	18.455534	356.674032	12.653063	22.532806
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148861	2.105710	8.707259	168.537116	2.164946	91.294864	7.141062	9.197104
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900000	1.129600	1.000000	187.000000	12.600000	0.320000	1.730000	5.000000
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025000	2.100175	4.000000	279.000000	17.400000	375.377500	6.950000	17.025000
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500000	3.207450	5.000000	330.000000	19.050000	391.440000	11.360000	21.200000
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075000	5.188425	24.000000	666.000000	20.200000	396.225000	16.955000	25.000000
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000000	12.126500	24.000000	711.000000	22.000000	396.900000	37.970000	50.000000

```
#check for null values
df.isnull().sum()
```

```
CRIM       0
ZN         0
INDUS      0
CHAS       0
NOX        0
RM         0
AGE        0
DIS        0
RAD        0
TAX        0
PTRATIO    0
B          0
LSTAT      0
Price      0
dtype: int64
```

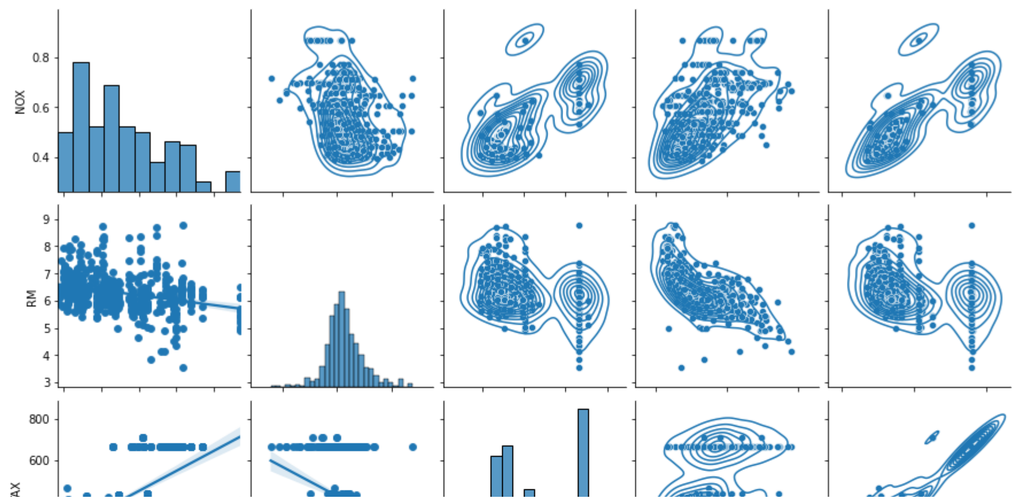
```
def create_vif(dataframe):
    ''' This function calculates the Variation Inflation Factors for each column and convert it into a dataframe'''

    #create an empty dataframe
    vif_table = pd.DataFrame()
    #populate the first column with the columns of the dataset
    vif_table['variables'] = dataframe.columns
    #calculate the VIF of each column and create a VIF column to store the number
    vif_table['VIF'] = [vif(dataframe.values, i) for i in range(df.shape[1])]

    return vif_table

#print the VIF table for each variable
print(create_vif(df))
```

```
   variables         VIF
0       CRIM    2.131404
1         ZN    2.910004
2      INDUS   14.485874
3       CHAS    1.176266
4        NOX   74.004269
5         RM  136.101743
6        AGE   21.398863
7        DIS   15.430455
8        RAD   15.369980
9        TAX   61.939713
10   PTRATIO   87.227233
11         B   21.351015
12     LSTAT   12.615188
13     Price   24.503206
```

```
#compress the columns 'DIS', 'RAD', 'INDUS' into 1 column
pca = PCA(n_components=1)
#call the compressed column 'new'
df['new'] = pca.fit_transform(df[['DIS', 'RAD', 'INDUS']])
#drop the three columns from the dataset
df = df.drop(['DIS', 'RAD', 'INDUS'], axis=1)
#recheck the new VIF table
print(create_vif(df))
```

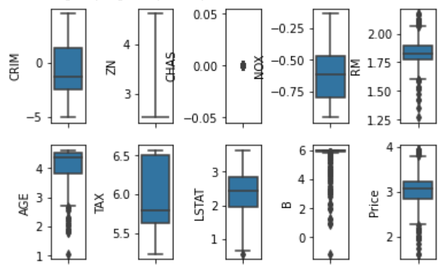
```
   variables         VIF
0       CRIM    2.006392
1         ZN    2.349186
2       CHAS    1.173519
3        NOX   65.166302
4         RM  133.757986
5        AGE   18.823276
6        TAX   56.391909
7    PTRATIO   77.938234
8          B   21.345554
9      LSTAT   12.580803
10     Price   23.131681
11       new    9.194328
```

```
#print a pairplot to check the relationships between strongly correlated features
pp = sns.pairplot(df[['NOX', 'RM', 'TAX', 'LSTAT', 'new']])
pp = pp.map_lower(sns.regplot)
pp = pp.map_upper(sns.kdeplot);
```




```
df1 = df.copy()
# # Create a figure with 10 subplots with a width spacing of 1.5
fig, ax = plt.subplots(2,5)
fig.subplots_adjust(wspace=1.5)

# Create a boxplot for the continuous features
box_plot1 = sns.boxplot(y=np.log(df1[df1.columns[0]]), ax=ax[0][0])
box_plot2 = sns.boxplot(y=np.log(df1[df1.columns[1]]), ax=ax[0][1])
box_plot3 = sns.boxplot(y=np.log(df1[df1.columns[2]]), ax=ax[0][2])
box_plot4 = sns.boxplot(y=np.log(df1[df1.columns[3]]), ax=ax[0][3])
box_plot5 = sns.boxplot(y=np.log(df1[df1.columns[4]]), ax=ax[0][4])
box_plot6 = sns.boxplot(y=np.log(df1[df1.columns[5]]), ax=ax[1][0])
box_plot7 = sns.boxplot(y=np.log(df1[df1.columns[6]]), ax=ax[1][1])
box_plot8 = sns.boxplot(y=np.log(df1[df1.columns[-3]]), ax=ax[1][2])
box_plot9 = sns.boxplot(y=np.log(df1[df1.columns[8]]), ax=ax[1][3])
box_plot10 = sns.boxplot(y=np.log(df1[df1.columns[10]]), ax=ax[1][4])
```

```
/usr/local/lib/python3.7/dist-packages/pandas/core/series.py:726: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
/usr/local/lib/python3.7/dist-packages/matplotlib/cbook/__init__.py:1157: RuntimeWarning: invalid value encountered in double_scalars
  notch_max = med + 1.57 * iqr / np.sqrt(N)
/usr/local/lib/python3.7/dist-packages/pandas/core/series.py:726: RuntimeWarning: divide by zero encountered in log
  result = getattr(ufunc, method)(*inputs, **kwargs)
/usr/local/lib/python3.7/dist-packages/matplotlib/cbook/__init__.py:1211: RuntimeWarning: invalid value encountered in double_scalars
  stats['iqr'] = q3 - q1
```




```
#One-Hot Encode the CHAS column
df = pd.get_dummies(df, columns=['CHAS'], drop_first=True)
#define the features and the labels, X and y
X = df.drop(['Price'], axis=1)
y = df['Price']

#split the features and labels into  train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)

#rescale the data to be robust to outliers
scaler = RobustScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)


#built the neural network architecture
model = Sequential()
model.add(Dense(15, input_dim=11, activation='relu'))
model.add(Dense(1, activation='linear'))
```

```
model.compile(loss='mse', optimizer='adam', metrics=['mse', 'mae'])


#train the neural network on the train dataset
history = model.fit(X_train, y_train, epochs=200, validation_split=0.2)
```

```
Epoch 1/200
11/11 [==============================] - 3s 17ms/step - loss: 561.5919 - mse: 561.5919 - mae: 21.6196 - val_loss: 534.3728 - val_mse: 534.3728 - val_mae: 20.9099
Epoch 2/200
11/11 [==============================] - 0s 4ms/step - loss: 552.2500 - mse: 552.2500 - mae: 21.3341 - val_loss: 525.9639 - val_mse: 525.9639 - val_mae: 20.6689
Epoch 3/200
11/11 [==============================] - 0s 4ms/step - loss: 543.4445 - mse: 543.4445 - mae: 21.0759 - val_loss: 517.5277 - val_mse: 517.5277 - val_mae: 20.4361
Epoch 4/200
11/11 [==============================] - 0s 4ms/step - loss: 534.7892 - mse: 534.7892 - mae: 20.8448 - val_loss: 509.0839 - val_mse: 509.0839 - val_mae: 20.2126
Epoch 5/200
11/11 [==============================] - 0s 4ms/step - loss: 526.1429 - mse: 526.1429 - mae: 20.6062 - val_loss: 501.2100 - val_mse: 501.2100 - val_mae: 20.0073
Epoch 6/200
11/11 [==============================] - 0s 4ms/step - loss: 517.9882 - mse: 517.9882 - mae: 20.4010 - val_loss: 492.8143 - val_mse: 492.8143 - val_mae: 19.7900
Epoch 7/200
11/11 [==============================] - 0s 4ms/step - loss: 509.7166 - mse: 509.7166 - mae: 20.2263 - val_loss: 484.7437 - val_mse: 484.7437 - val_mae: 19.5879
Epoch 8/200
11/11 [==============================] - 0s 4ms/step - loss: 501.6067 - mse: 501.6067 - mae: 20.0618 - val_loss: 476.6565 - val_mse: 476.6565 - val_mae: 19.3925
Epoch 9/200
11/11 [==============================] - 0s 4ms/step - loss: 493.5188 - mse: 493.5188 - mae: 19.9086 - val_loss: 468.7867 - val_mse: 468.7867 - val_mae: 19.2202
Epoch 10/200
11/11 [==============================] - 0s 4ms/step - loss: 485.5971 - mse: 485.5971 - mae: 19.7506 - val_loss: 461.6504 - val_mse: 461.6504 - val_mae: 19.0809
Epoch 11/200
11/11 [==============================] - 0s 4ms/step - loss: 477.3806 - mse: 477.3806 - mae: 19.5912 - val_loss: 453.8126 - val_mse: 453.8126 - val_mae: 18.9127
Epoch 12/200
11/11 [==============================] - 0s 4ms/step - loss: 469.0844 - mse: 469.0844 - mae: 19.4246 - val_loss: 445.5368 - val_mse: 445.5368 - val_mae: 18.7452
Epoch 13/200
11/11 [==============================] - 0s 4ms/step - loss: 460.3438 - mse: 460.3438 - mae: 19.2428 - val_loss: 437.9861 - val_mse: 437.9861 - val_mae: 18.6005
Epoch 14/200
11/11 [==============================] - 0s 4ms/step - loss: 451.5661 - mse: 451.5661 - mae: 19.0546 - val_loss: 429.8988 - val_mse: 429.8988 - val_mae: 18.4390
Epoch 15/200
11/11 [==============================] - 0s 4ms/step - loss: 442.0199 - mse: 442.0199 - mae: 18.8521 - val_loss: 420.9161 - val_mse: 420.9161 - val_mae: 18.2540
Epoch 16/200
11/11 [==============================] - 0s 4ms/step - loss: 432.2784 - mse: 432.2784 - mae: 18.6506 - val_loss: 411.5705 - val_mse: 411.5705 - val_mae: 18.0556
Epoch 17/200
11/11 [==============================] - 0s 4ms/step - loss: 422.3843 - mse: 422.3843 - mae: 18.4460 - val_loss: 401.5641 - val_mse: 401.5641 - val_mae: 17.8508
Epoch 18/200
11/11 [==============================] - 0s 4ms/step - loss: 411.6464 - mse: 411.6464 - mae: 18.2075 - val_loss: 392.5245 - val_mse: 392.5245 - val_mae: 17.6623
Epoch 19/200
11/11 [==============================] - 0s 4ms/step - loss: 401.9130 - mse: 401.9130 - mae: 17.9780 - val_loss: 383.2000 - val_mse: 383.2000 - val_mae: 17.4619
Epoch 20/200
11/11 [==============================] - 0s 4ms/step - loss: 391.4576 - mse: 391.4576 - mae: 17.7404 - val_loss: 373.3798 - val_mse: 373.3798 - val_mae: 17.2450
Epoch 21/200
11/11 [==============================] - 0s 4ms/step - loss: 380.8622 - mse: 380.8622 - mae: 17.4891 - val_loss: 363.2141 - val_mse: 363.2141 - val_mae: 17.0187
Epoch 22/200
11/11 [==============================] - 0s 4ms/step - loss: 369.3805 - mse: 369.3805 - mae: 17.2194 - val_loss: 353.3207 - val_mse: 353.3207 - val_mae: 16.7901
Epoch 23/200
11/11 [==============================] - 0s 4ms/step - loss: 358.6265 - mse: 358.6265 - mae: 16.9619 - val_loss: 343.0705 - val_mse: 343.0705 - val_mae: 16.5522
Epoch 24/200
11/11 [==============================] - 0s 4ms/step - loss: 347.7412 - mse: 347.7412 - mae: 16.6699 - val_loss: 333.4461 - val_mse: 333.4461 - val_mae: 16.3228
Epoch 25/200
11/11 [==============================] - 0s 4ms/step - loss: 336.6387 - mse: 336.6387 - mae: 16.3763 - val_loss: 323.3958 - val_mse: 323.3958 - val_mae: 16.0772
Epoch 26/200
11/11 [==============================] - 0s 4ms/step - loss: 325.7775 - mse: 325.7775 - mae: 16.0878 - val_loss: 313.2889 - val_mse: 313.2889 - val_mae: 15.8227
Epoch 27/200
11/11 [==============================] - 0s 4ms/step - loss: 314.7567 - mse: 314.7567 - mae: 15.7843 - val_loss: 303.4294 - val_mse: 303.4294 - val_mae: 15.5661
Epoch 28/200
11/11 [==============================] - 0s 4ms/step - loss: 303.6806 - mse: 303.6806 - mae: 15.4703 - val_loss: 293.4657 - val_mse: 293.4657 - val_mae: 15.2987
Epoch 29/200
11/11 [==============================] - 0s 4ms/step - loss: 293.1223 - mse: 293.1223 - mae: 15.1560 - val_loss: 283.7514 - val_mse: 283.7514 - val_mae: 15.0296
Epoch 30/200
```

```
#plot the loss and validation loss of the dataset
history_df = pd.DataFrame(history.history)
plt.plot(history_df['loss'], label='loss')
plt.plot(history_df['val_loss'], label='val_loss')

plt.legend()
```

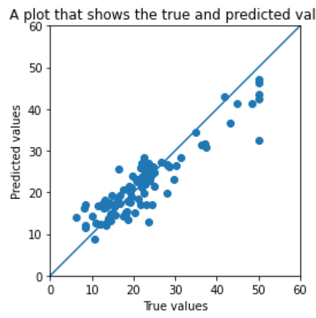
```
<matplotlib.legend.Legend at 0x7fbff2141a50>
```

```
#evaluate the model
model.evaluate(X_test, y_test, batch_size=128)
```

```
1/1 [==============================] - 0s 15ms/step - loss: 17.5405 - mse: 17.5405 - mae: 3.1404
[17.54050064086914, 17.54050064086914, 3.1403918266296387]
```

```
y_pred = model.predict(X_test).flatten()

a = plt.axes(aspect='equal')
plt.scatter(y_test, y_pred)
plt.xlabel('True values')
plt.ylabel('Predicted values')
plt.title('A plot that shows the true and predicted values')
plt.xlim([0, 60])
plt.ylim([0, 60])
plt.plot([0, 60], [0, 60])
```

```
[<matplotlib.lines.Line2D at 0x7fbff202c5d0>]
```

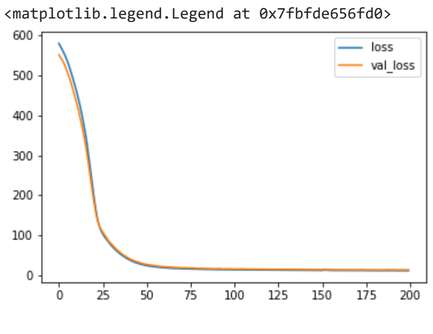



```
#built the neural network architecture
model = Sequential()
model.add(Dense(15, input_dim=11, activation='relu'))
model.add(Dense(7, activation='relu'))
model.add(Dense(3, activation='relu'))
model.add(Dense(1, activation='linear'))

model.compile(loss='mse', optimizer='adam', metrics=['mse', 'mae'])

#train the neural network on the train dataset
history = model.fit(X_train, y_train, epochs=200, validation_split=0.2)
```

```
Epoch 1/200
11/11 [==============================] - 1s 14ms/step - loss: 579.6306 - mse: 579.6306 - mae: 22.1838 - val_loss: 551.3060 - val_mse: 551.3060 - val_mae: 21.3288
Epoch 2/200
11/11 [==============================] - 0s 4ms/step - loss: 571.4150 - mse: 571.4150 - mae: 21.9162 - val_loss: 544.2689 - val_mse: 544.2689 - val_mae: 21.1288
Epoch 3/200
11/11 [==============================] - 0s 4ms/step - loss: 563.1451 - mse: 563.1451 - mae: 21.6395 - val_loss: 536.9783 - val_mse: 536.9783 - val_mae: 20.9219
Epoch 4/200
11/11 [==============================] - 0s 4ms/step - loss: 554.2074 - mse: 554.2074 - mae: 21.3624 - val_loss: 528.8367 - val_mse: 528.8367 - val_mae: 20.7059
Epoch 5/200
11/11 [==============================] - 0s 4ms/step - loss: 544.3029 - mse: 544.3029 - mae: 21.0605 - val_loss: 518.2978 - val_mse: 518.2978 - val_mae: 20.4320
Epoch 6/200
11/11 [==============================] - 0s 4ms/step - loss: 533.1375 - mse: 533.1375 - mae: 20.7480 - val_loss: 506.3035 - val_mse: 506.3035 - val_mae: 20.1089
Epoch 7/200
11/11 [==============================] - 0s 4ms/step - loss: 520.4274 - mse: 520.4274 - mae: 20.3939 - val_loss: 493.3994 - val_mse: 493.3994 - val_mae: 19.7581
Epoch 8/200
11/11 [==============================] - 0s 4ms/step - loss: 506.5990 - mse: 506.5990 - mae: 20.0253 - val_loss: 479.4516 - val_mse: 479.4516 - val_mae: 19.4042
Epoch 9/200
11/11 [==============================] - 0s 4ms/step - loss: 491.8128 - mse: 491.8128 - mae: 19.6552 - val_loss: 464.3579 - val_mse: 464.3579 - val_mae: 19.0072
Epoch 10/200
11/11 [==============================] - 0s 4ms/step - loss: 476.4883 - mse: 476.4883 - mae: 19.3120 - val_loss: 448.5938 - val_mse: 448.5938 - val_mae: 18.5920
Epoch 11/200
11/11 [==============================] - 0s 4ms/step - loss: 460.6993 - mse: 460.6993 - mae: 18.9799 - val_loss: 432.2875 - val_mse: 432.2875 - val_mae: 18.2051
Epoch 12/200
11/11 [==============================] - 0s 4ms/step - loss: 443.1344 - mse: 443.1344 - mae: 18.6020 - val_loss: 414.9423 - val_mse: 414.9423 - val_mae: 17.8084
Epoch 13/200
11/11 [==============================] - 0s 4ms/step - loss: 424.7203 - mse: 424.7203 - mae: 18.2162 - val_loss: 395.8308 - val_mse: 395.8308 - val_mae: 17.3565
Epoch 14/200
11/11 [==============================] - 0s 4ms/step - loss: 403.8983 - mse: 403.8983 - mae: 17.7999 - val_loss: 375.8900 - val_mse: 375.8900 - val_mae: 16.8616
Epoch 15/200
11/11 [==============================] - 0s 4ms/step - loss: 381.1607 - mse: 381.1607 - mae: 17.2884 - val_loss: 354.2267 - val_mse: 354.2267 - val_mae: 16.3044
Epoch 16/200
11/11 [==============================] - 0s 4ms/step - loss: 357.0889 - mse: 357.0889 - mae: 16.7282 - val_loss: 330.2659 - val_mse: 330.2659 - val_mae: 15.6547
Epoch 17/200
11/11 [==============================] - 0s 4ms/step - loss: 328.4629 - mse: 328.4629 - mae: 16.0029 - val_loss: 303.5121 - val_mse: 303.5121 - val_mae: 14.9393
Epoch 18/200
11/11 [==============================] - 0s 4ms/step - loss: 296.4904 - mse: 296.4904 - mae: 15.1441 - val_loss: 273.4820 - val_mse: 273.4820 - val_mae: 14.0963
Epoch 19/200
11/11 [==============================] - 0s 4ms/step - loss: 262.5279 - mse: 262.5279 - mae: 14.1630 - val_loss: 242.1958 - val_mse: 242.1958 - val_mae: 13.1042
Epoch 20/200
11/11 [==============================] - 0s 4ms/step - loss: 227.9474 - mse: 227.9474 - mae: 13.0177 - val_loss: 210.2778 - val_mse: 210.2778 - val_mae: 12.0419
Epoch 21/200
11/11 [==============================] - 0s 4ms/step - loss: 193.8318 - mse: 193.8318 - mae: 11.8113 - val_loss: 181.6446 - val_mse: 181.6446 - val_mae: 11.1757
Epoch 22/200
11/11 [==============================] - 0s 4ms/step - loss: 164.3643 - mse: 164.3643 - mae: 10.6382 - val_loss: 156.9065 - val_mse: 156.9065 - val_mae: 10.3726
Epoch 23/200
11/11 [==============================] - 0s 4ms/step - loss: 139.5271 - mse: 139.5271 - mae: 9.7138 - val_loss: 137.1810 - val_mse: 137.1810 - val_mae: 9.6214
Epoch 24/200
11/11 [==============================] - 0s 4ms/step - loss: 123.0435 - mse: 123.0435 - mae: 9.0431 - val_loss: 123.9564 - val_mse: 123.9564 - val_mae: 9.0035
Epoch 25/200
11/11 [==============================] - 0s 4ms/step - loss: 111.0643 - mse: 111.0643 - mae: 8.5125 - val_loss: 114.9398 - val_mse: 114.9398 - val_mae: 8.5596
Epoch 26/200
11/11 [==============================] - 0s 4ms/step - loss: 103.4120 - mse: 103.4120 - mae: 8.1434 - val_loss: 107.5756 - val_mse: 107.5756 - val_mae: 8.2342
Epoch 27/200
11/11 [==============================] - 0s 4ms/step - loss: 96.1752 - mse: 96.1752 - mae: 7.8239 - val_loss: 100.9334 - val_mse: 100.9334 - val_mae: 7.9564
Epoch 28/200
11/11 [==============================] - 0s 4ms/step - loss: 90.3110 - mse: 90.3110 - mae: 7.5599 - val_loss: 94.4981 - val_mse: 94.4981 - val_mae: 7.6545
Epoch 29/200
11/11 [==============================] - 0s 4ms/step - loss: 84.5785 - mse: 84.5785 - mae: 7.2804 - val_loss: 87.8842 - val_mse: 87.8842 - val_mae: 7.3444
Epoch 30/200
```

```
#plot the loss and validation loss of the dataset
history_df = pd.DataFrame(history.history)
plt.plot(history_df['loss'], label='loss')
plt.plot(history_df['val_loss'], label='val_loss')

plt.legend()
```

```
<matplotlib.legend.Legend at 0x7fbfde656fd0>
```




```
#evaluate the model
model.evaluate(X_test, y_test, batch_size=128)
```

```
1/1 [==============================] - 0s 15ms/step - loss: 14.0964 - mse: 14.0964 - mae: 2.8246
[14.09637451171875, 14.09637451171875, 2.8245575428009033]
```

```
y_pred = model.predict(X_test).flatten()

a = plt.axes(aspect='equal')
plt.scatter(y_test, y_pred)
plt.xlabel('True values')
plt.ylabel('Predicted values')
plt.title('A plot that shows the true and predicted values')
plt.xlim([0, 60])
plt.ylim([0, 60])
plt.plot([0, 60], [0, 60])
```

```
[<matplotlib.lines.Line2D at 0x7fbfde5afed0>]
```