In [ ]:
```
!pip install tensorflow
```

In [ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from keras.models import Sequential
from keras.layers import Dense
import matplotlib.pyplot as plt

df = pd.read_csv('BostonHousing.csv')
df.head(n=10)
```

In [ ]:
```
# df.drop(columns=['CAT. MEDV'],inplace=True)
# df.dropna(inplace=True)
df.isnull().sum()
```

Out[ ]:
```
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
medv       0
dtype: int64
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 #   Column   Non-Null Count  Dtype  
---  ------   --------------  -----  
 0   crim     506 non-null    float64
 1   zn       506 non-null    float64
 2   indus    506 non-null    float64
 3   chas     506 non-null    int64  
 4   nox      506 non-null    float64
 5   rm       506 non-null    float64
 6   age      506 non-null    float64
 7   dis      506 non-null    float64
 8   rad      506 non-null    int64  
 9   tax      506 non-null    int64  
 10  ptratio  506 non-null    float64
 11  b        506 non-null    float64
 12  lstat    506 non-null    float64
 13  medv     506 non-null    float64
dtypes: float64(11), int64(3)
memory usage: 55.5 KB
```

In [ ]:
```
df.describe()
```

Out[ ]:

	crim	zn	indus	chas	nox	rm	a
count	506.000000	506.000000	506.000000	506.000000	506.000000	506.000000	506.000(
mean	3.613524	11.363636	11.136779	0.069170	0.554695	6.284634	68.574(
std	8.601545	23.322453	6.860353	0.253994	0.115878	0.702617	28.148(
min	0.006320	0.000000	0.460000	0.000000	0.385000	3.561000	2.900(
25%	0.082045	0.000000	5.190000	0.000000	0.449000	5.885500	45.025(
50%	0.256510	0.000000	9.690000	0.000000	0.538000	6.208500	77.500(
75%	3.677083	12.500000	18.100000	0.000000	0.624000	6.623500	94.075(
max	88.976200	100.000000	27.740000	1.000000	0.871000	8.780000	100.000(

In [ ]:
```
df.corr()['medv'].sort_values()
```

Out[ ]:
```
lstat     -0.737663
ptratio   -0.507787
indus     -0.483725
tax       -0.468536
nox       -0.427321
crim      -0.388305
rad       -0.381626
age       -0.376955
chas       0.175260
dis        0.249929
b          0.333461
zn         0.360445
rm         0.695360
medv       1.000000
Name: medv, dtype: float64
```

In [ ]:
```
X = df.loc[:, df.columns != 'medv'].values  #or X = df.loc[:,['lstat','ptratio',
y = df.loc[:, df.columns == 'medv'].values  #or y = df.loc[:,'medv']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random
```

Normalizing Training and Testing Data Set

In [ ]:
```
scaler = StandardScaler() #standardise (Z-score normalization => mean = 0 and St
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

input_shape=(13,): This parameter defines the shape of the input data for the first layer. In this case, it expects input data with 13 features.

activation='relu': This parameter specifies the activation function to be used in the layer. Activation functions introduce non-linearity to the network, enabling it to learn complex patterns. In this case, the Rectified Linear Unit (ReLU) activation function is used, which returns the input if it is positive and 0 otherwise.

optimizer='adam': This parameter specifies the optimization algorithm used during training. Adam (Adaptive Moment Estimation) is a popular optimization algorithm known for its efficiency and effectiveness in a wide range of deep learning tasks.

loss='mse': This parameter defines the loss function used to measure the difference between the predicted output and the true output during training. MSE (Mean Squared Error) is a common loss function for regression tasks that calculates the average squared difference between the predicted and true values.

metrics=['mae']: This parameter specifies the evaluation metric(s) used to monitor the model's performance during training. MAE (Mean Absolute Error) is a metric that measures the average absolute difference between the predicted and true values. It provides a measure of the model's accuracy.

`Adam optimizer dynamically adjusts the learning rates for individual parameters in a neural network based on the history of gradients. This adaptive learning rate strategy helps improve the efficiency and robustness of the optimization process during training.`

In [ ]:
```
model = Sequential(layers=[Dense(128, input_shape=(13, ), activation='relu', nam


# model.add(Dense(128, input_shape=(13, ), activation='relu', name='dense_1'))
# model.add(Dense(64, activation='relu', name='dense_2'))
# model.add(Dense(1, activation='linear', name='dense_output'))

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

```
Model: "sequential_14"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_1 (Dense)             (None, 128)               1792

 dense_2 (Dense)             (None, 64)                8256

 dense_output (Dense)        (None, 1)                 65

=================================================================
Total params: 10113 (39.50 KB)
Trainable params: 10113 (39.50 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [ ]:
```
model.fit(X_train, y_train, epochs=100, validation_split=0.05, verbose = 'auto')
```

Out[ ]: <keras.src.callbacks.History at 0x29a83ca10>

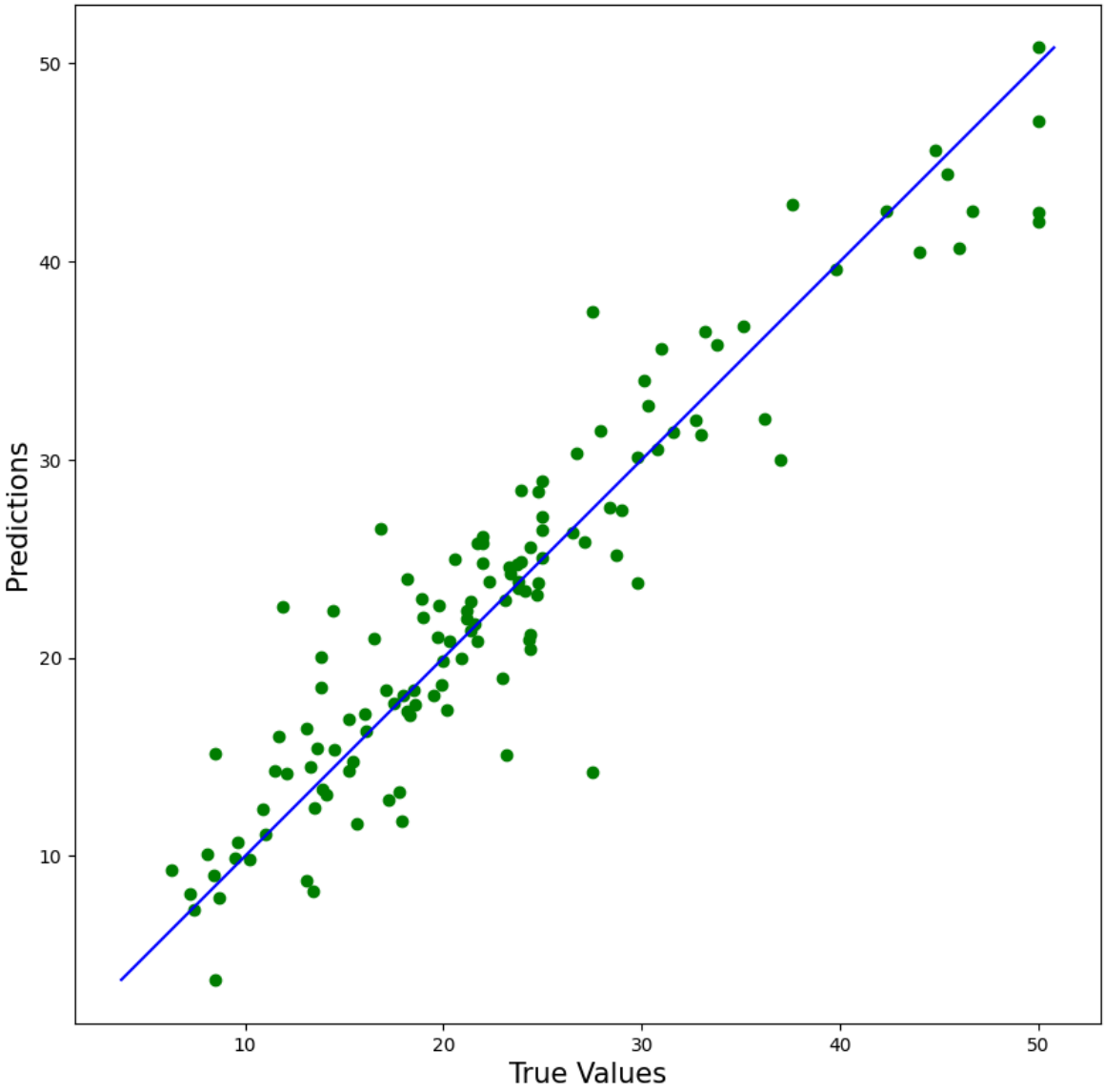
In [ ]:
```
from sklearn.metrics import r2_score
y_pred = model.predict(X_test)
mse_nn, mae_nn= model.evaluate(X_test, y_test)
r2 = r2_score(y_test, y_pred)

print('Mean squared error on test data: ', mse_nn)
print('Mean absolute error on test data: ', mae_nn)
print('Accuracy:', r2*100)
```

```
4/4 [==============================] - 0s 543us/step
4/4 [==============================] - 0s 711us/step - loss: 13.2702 - mae: 2.641
8
Mean squared error on test data:  13.270233154296875
Mean absolute error on test data:  2.6417903900146484
Accuracy: 86.903562213009
```

In [ ]:
```
plt.figure(figsize=(10,10))
plt.scatter(y_test, y_pred, c='green')

p1 = max(max(y_pred), max(y_test))
p2 = min(min(y_pred), min(y_test))
plt.plot([p1, p2], [p1, p2], 'b-')
plt.xlabel('True Values', fontsize=15)
plt.ylabel('Predictions', fontsize=15)
plt.axis('equal')
plt.show()
```



In [ ]:

In [ ]: