MiniProject 3 - ItDS

Ken Erikson

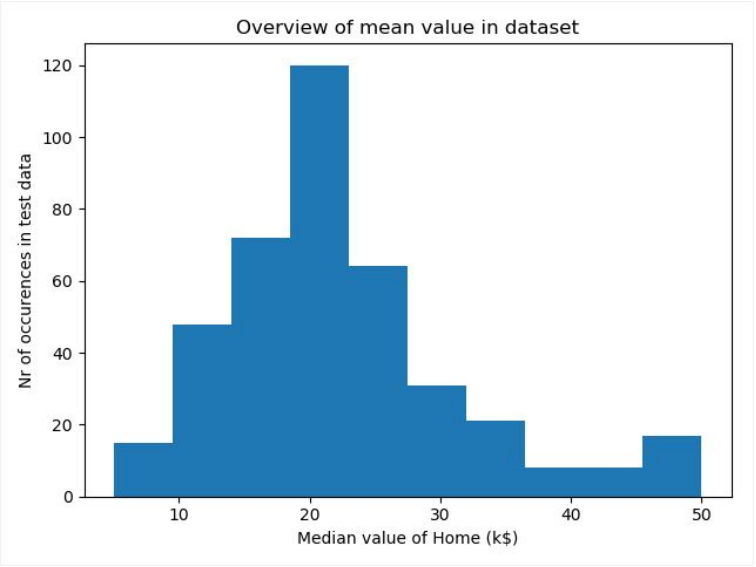
Problem:
Implement a basic deep learning model that predicts ‘MEDV - Median value of owner-occupied homes in $1000's’ using Tensorflow (keras) and the boston housing set.

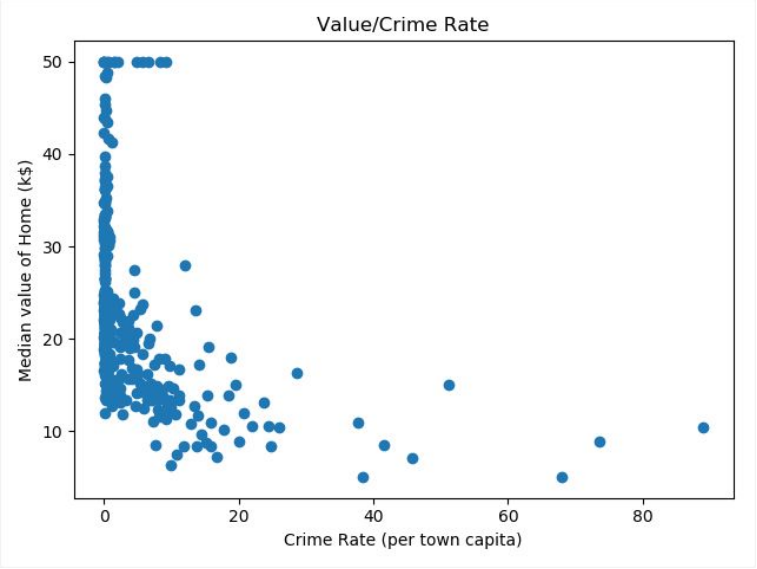
1. Provide descriptive statistics about the housing dataset?

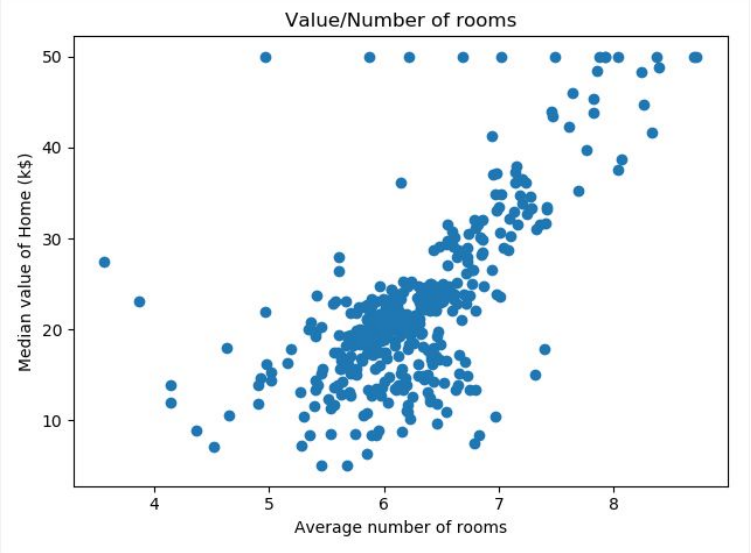
```
print("The train input is a tuple of shape: "+str(x_train.shape))
print("The train output is an array of length: "+str(len(y_train)))

median_value_array = y_train
plt.title("Overview of mean value in dataset")
plt.xlabel("Median value of Home (k$)")
plt.ylabel("Nr of occurrences in test data")
plt.hist(median_value_array)
plt.show()

crim_rate = np.array(x_train[:,0])
plt.title("Value/Crime Rate")
plt.xlabel("Crime Rate (per town capita)")
plt.ylabel("Median value of Home (k$)")
plt.scatter(crim_rate,median_value_array,linestyle = 'None')
plt.show()

average_number_of_rooms = np.array(x_train[:,5])
plt.title("Value/Number of rooms")
plt.xlabel("Average number of rooms")
plt.ylabel("Median value of Home (k$)")
plt.scatter(average_number_of_rooms,median_value_array,linestyle = 'None')
plt.show()
```

Example prints and Plots:

```
The train input is a tuple of shape: (404, 13)
The train output is an array of length: 404
```


Overview of mean value in dataset
Nr of occurences in test data
Median value of Home (k$)
0
20
40
60
80
100
120
10
20
30
40
50


Value/Crime Rate
Median value of Home (k$)
Crime Rate (per town capita)
10
20
30
40
50
0
20
40
60
80


Value/Number of rooms
Median value of Home (k$)
Average number of rooms
10
20
30
40
50
4
5
6
7
8

2. Use deep learning approaches to build a predictive model where the output is “MEDV - Median value of owner-occupied homes in $1000's”

2.1 Define your model architecture and explain how you have decided on choosing this architecture

I have chosen a 2 layer setup (not-including input and output layer). I got the layer numbers through manual and automatic random testing:

(Here 0-4 extra layers with every layer 4-20 nodes)

```
scale_int = random.randint(0, 100)
amountOfLayers = 0
if (scale_int < 50):
   amountOfLayers = 1
elif (scale_int < 65):
   amountOfLayers = 2
elif (scale_int < 80):
   amountOfLayers = 3
elif (scale_int < 90):
   amountOfLayers = 4
layerToAdd = 1

layer_sizes=[random.randint(4,20) for _ in range(amountOfLayers+1)]
model = keras.Sequential()
model.add(keras.layers.Dense(layer_sizes[0], activation=tf.nn.relu,
kernel_initializer='normal', input_shape=(13,)))
# model.add(keras.layers.Dropout(0.2)) # Not Overfitting data
while (layerToAdd <= amountOfLayers):
   layerToAdd = layerToAdd + 1
   model.add(keras.layers.Dense(layer_sizes[1], kernel_initializer='normal',
activation=tf.nn.relu))
model.add(keras.layers.Dense(1, kernel_initializer='normal'))
model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='mean_squared_error')
```

The Result after some testing:

```
model = keras.Sequential()
model.add(keras.layers.Dense(9,
activation=tf.nn.relu,kernel_initializer='normal', input_shape=(13,)))
#model.add(keras.layers.Dropout(0.2)) # Not Overfitting data
model.add(keras.layers.Dense(11, kernel_initializer='normal',
activation=tf.nn.relu))
model.add(keras.layers.Dense(1, kernel_initializer='normal'))
```

2.2. Compile the model

Using AdamOptimizer as a sane default.

```
model.compile(optimizer=tf.train.AdamOptimizer(),
              loss='mean_squared_error')
```

2.3 Divide the data to training and test set and fit your model on the training data

```
(x_train, y_train), (x_test, y_test) = keras.datasets.boston_housing.load_data()
```

```
model.fit(x_train, y_train, epochs=2000, verbose=0)
```

2.4 Evaluate your model on the test set

```
mean_squared_error = model.evaluate(x_test, y_test)
print("Mean squared error on test data: "+str(round(mean_squared_error,2))+"k$")
print("Mean error on test data: "+str(round(sqrt(mean_squared_error),2))+"k$")
```

Example print:

```
Mean squared error on test data: 23.34k$
Mean error on test data: 4.83k$
```