



ChatGPT 3.5 ▾



You  
#!/usr/bin/env python  
# coding: utf-8

# In[1]:

```
import pandas as pd  
df = pd.read_csv("./1_boston_housing.csv")
```

# In[2]:

df

# In[3]:

```
df.isnull().sum()
```



# In[4]:

```
from sklearn.model_selection import train_test_split  
  
x = df.loc[:,df.columns != 'MEDV']  
y = df.loc[:,df.columns == 'MEDV']  
  
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.3,random_state=1)
```

# In[5]:

```
from sklearn.preprocessing import MinMaxScaler  
mms = MinMaxScaler()  
x_train = mms.fit_transform(x_train)  
x_test = mms.transform(x_test)
```



# In[6]:

```
from keras.models import Sequential  
from keras.layers import Dense  
  
model = Sequential()  
  
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()
```



# In[7]:

```
history = model.fit(x_train,y_train,epochs=100,validation_split=0.05,verbose=1)
```

# In[8]:

```

}                               mse,mae = model.evaluate(x_test,y_test)
                                print('MSE : ',mse)
                                print('MAE : ',mae)

# In[9]:
```

↓

```

y1 = model.predict(x_test[:])
```

↓

```

# In[10]:
```

```

y_test
```

```

# In[11]:
```

}

```

ps=[]
for i in y1:
    ps.append(list(i)[0])

# In[12]:
```

↓

```

d = pd.DataFrame({'actual':y_test['MEDV'],'predicted':ps})
```

```

# In[13]:
```

```

d
```

tell me what each cell is doing in this assignment , what is the objective what are the results

}

 ChatGPT

Sure, let's break down each cell in the provided Python code:

- Cell 1:**
  - It starts with a shebang line `#!/usr/bin/env python` which is used to indicate to the system that this script should be executed using the Python interpreter.
  - It sets the encoding of the script to UTF-8 to handle Unicode characters properly.
  - It imports the pandas library as `pd`.
  - It reads a CSV file named "l\_boston\_housing.csv" into a DataFrame `df` using `pd.read\_csv()`.
- Cell 2:**
  - Displays the DataFrame `df`, showing the contents of the dataset loaded from the CSV file.
- Cell 3:**
  - Checks for any missing values in the DataFrame `df` using the `isnull()` method followed by `sum()` to count the missing values for each column.
- Cell 4:**
  - Splits the data into features (X) and the target variable (y) for the machine learning model.
  - Uses `train\_test\_split` from `sklearn.model\_selection` to split the data into training and testing sets with a 70-30 split ratio.
- Cell 5:**
  - Imports `MinMaxScaler` from `sklearn.preprocessing` to scale the features to a range between 0 and 1.
  - Applies Min-Max scaling to the training and testing features separately.
- Cell 6:**

↓

  - Sets up a sequential neural network model using Keras.
  - Adds three dense layers with 128, 64, and 1 neuron(s) respectively. They use ReLU activation

functions except for the output layer, which uses linear activation.

- Compiles the model with the Adam optimizer, mean squared error (MSE) loss function, and mean absolute error (MAE) metric.
- Displays the summary of the model architecture.

#### 7. Cell 7:

- Fits the model to the training data for 100 epochs with a validation split of 5%.
- The training process is verbose, meaning it displays progress information during training.

#### 8. Cell 8:

- Evaluates the trained model on the test data and computes the MSE and MAE metrics.

#### 9. Cell 9:

- Uses the trained model to predict the target variable (housing prices) for the test features.

#### 10. Cell 10:

- Displays the actual and predicted values of the target variable ('MEDV') from the test dataset.

#### 11. Cell 11:

- Computes the actual and predicted values of the target variable and stores them in a DataFrame 'd'.

The objective of this assignment seems to be to develop a neural network model to predict housing prices (MEDV) based on various features in the Boston housing dataset. The results include the model's evaluation metrics (MSE and MAE) and the comparison between actual and predicted housing prices.



You

tell me all evaluation metrics



ChatGPT

In this code snippet, the evaluation metric used is Mean Squared Error (MSE) and Mean Absolute Error (MAE). Let's define them:

#### 1. Mean Squared Error (MSE):

- MSE is a measure of the average squared difference between the actual and predicted values.
- It is calculated by taking the average of the squared differences between the predicted values and the actual values.
- Mathematically, MSE is calculated as:  
$$\text{MSE} = \frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$
where  $n$  is the number of samples,  $y_i$  is the actual value, and  $\hat{y}_i$  is the predicted value.

#### 2. Mean Absolute Error (MAE):

- MAE is a measure of the average absolute difference between the actual and predicted values.
- It is calculated by taking the average of the absolute differences between the predicted values and the actual values.
- Mathematically, MAE is calculated as:  
$$\text{MAE} = \frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$
where  $n$  is the number of samples,  $y_i$  is the actual value, and  $\hat{y}_i$  is the predicted value.

In the provided code, after training the neural network model, it evaluates the model's performance on the test data using both MSE and MAE. These metrics give insights into how well the model is performing in terms of predicting housing prices compared to the actual values.



You

what we are trying to achieve with this code what is boston housing data set and how this code helps to get



ChatGPT

The objective of the code is to build a machine learning model that can predict housing prices based on various features using the Boston Housing dataset. Let's break it down:

## 1. BOSTON HOUSING DATASET

- The Boston Housing dataset is a widely used dataset in machine learning. It contains information collected by the U.S Census Service concerning housing in the area of Boston, Massachusetts.
- This dataset is often used for regression tasks, where the goal is to predict a continuous target variable, in this case, the median value of owner-occupied homes (MEDV), based on other features.



### 2. Objective:

- The primary goal of this code is to train a neural network model to accurately predict the median value of owner-occupied homes (MEDV) based on a set of features such as crime rate, number of rooms, pupil-teacher ratio, etc.
- By analyzing the relationships between these features and the target variable (housing prices), the model aims to learn patterns and make predictions on unseen data.

### 3. How the Code Helps:

- The code loads the Boston Housing dataset and preprocesses it by splitting it into training and testing sets and scaling the features to a uniform range using Min-Max scaling.
- It then builds a neural network model using the Keras library, which is a high-level neural networks API running on top of TensorFlow or Theano.
- The model architecture consists of multiple layers of neurons, with the final layer predicting the housing prices.
- The model is trained on the training data using the Mean Squared Error (MSE) loss function and optimized using the Adam optimizer.
- After training, the model is evaluated using both MSE and Mean Absolute Error (MAE) on the test data to assess its performance.
- Finally, the model's predictions on the test data are compared with the actual values to understand how well the model generalizes to unseen data.



In summary, this code aims to create a machine learning model that can accurately predict housing prices based on historical data, which could be valuable for various applications such as real estate market analysis, property valuation, or investment decisions.



You

explain in detail how it is doing tha



ChatGPT

Certainly! Let's break down how the code achieves its objective of predicting housing prices using the Boston Housing dataset:

### 1. Data Loading and Preprocessing:

- The code begins by loading the Boston Housing dataset from a CSV file using the `pd.read\_csv()` function from the pandas library. This dataset contains various features (e.g., crime rate, number of rooms, etc.) and the target variable, which is the median value of owner-occupied homes (MEDV).
- After loading the dataset, the code performs a check for missing values using `df.isnull().sum()` to ensure data quality.
- The dataset is then split into features (X) and the target variable (y) using `train\_test\_split` from `sklearn.model\_selection`.

### 2. Feature Scaling:

- To ensure that all features are on a similar scale, the code applies Min-Max scaling using `MinMaxScaler` from `sklearn.preprocessing`. This scaling technique transforms each feature to a given range (by default, between 0 and 1).

### 3. Neural Network Model Construction:

- The code constructs a neural network model using Keras, a high-level neural networks API.
- The model is built sequentially using `Sequential()` from `keras.models`.
- It adds three dense (fully connected) layers using `Dense` from `keras.layers`. The first two layers use the ReLU activation function, and the output layer uses a linear activation function since it's a regression task.
- The model is compiled with the Adam optimizer, Mean Squared Error (MSE) loss function, and Mean Absolute Error (MAE) metric using `compile()`.

### 4. Model Training:

- The model is trained on the training data using the `fit()` method. It runs for 100 epochs with a validation split of 5%(`validation\_split=0.05`). During training, the model learns to minimize the MSE loss function using backpropagation.

#### 5. Model Evaluation:

- After training, the model's performance is evaluated on the test data using both MSE and MAE metrics. This is done using the `evaluate()` method.

#### 6. Prediction:

- Finally, the trained model is used to make predictions on the test data using the `predict()` method.  


#### 7. Comparison of Predictions and Actual Values:

- The code compares the actual values of the target variable (`MEDV`) from the test dataset with the predicted values obtained from the model.
- These actual and predicted values are stored in a DataFrame for analysis.

By following these steps, the code builds, trains, and evaluates a neural network model to predict housing prices based on the features provided in the Boston Housing dataset. It leverages the power of neural networks to learn complex relationships between the input features and the target variable, ultimately providing predictions that can be useful for various real-world applications.



You

explain all features in boston housing



ChatGPT

The Boston Housing dataset contains several features that are potentially related to housing prices. Here's an explanation of each feature:

##### 1. CRIM (Per capita crime rate by town):

- This feature represents the per capita crime rate in the town. Higher values indicate a higher crime rate in the area.

##### 2. ZN (Proportion of residential land zoned for lots over 25,000 sq. ft.):

- ZN represents the proportion of residential land zoned for large lots, typically over 25,000 square feet. It indicates the degree of residential zoning in the area.

##### 3. INDUS (Proportion of non-retail business acres per town):

- INDUS represents the proportion of non-retail business acres per town. It indicates the industrialization level of the area.

##### 4. CHAS (Charles River dummy variable: 1 if tract bounds river; 0 otherwise):

- CHAS is a binary variable indicating whether the housing tract is bounded by the Charles River. A value of 1 indicates that the tract bounds the river, while 0 indicates otherwise.

##### 5. NOX (Nitric oxides concentration, parts per 10 million):

- NOX represents the concentration of nitric oxides (NOX) in parts per 10 million in the air. It indicates the level of air pollution in the area.

##### 6. RM (Average number of rooms per dwelling):

- RM represents the average number of rooms per dwelling (house) in the area. It provides an indication of the size of the houses in the neighborhood.

##### 7. AGE (Proportion of owner-occupied units built before 1940):

- AGE represents the proportion of owner-occupied units built before 1940. It gives an idea of the age distribution of houses in the area.

##### 8. DIS (Weighted distances to five Boston employment centers):

- DIS represents the weighted distances to five employment centers in Boston. It indicates the accessibility of employment opportunities from the neighborhood.

##### 9. RAD (Index of accessibility to radial highways):

- RAD represents the index of accessibility to radial highways. It measures the ease of access to radial highways from the location.

##### 10. TAX (Full-value property tax rate per \$10,000):

- TAX represents the full-value property tax rate per \$10,000 of property value. It indicates the property tax burden in the area.

##### 11. PTRATIO (Pupil-teacher ratio by town):

- PTRATIO represents the pupil-teacher ratio by town. It measures the number of students per teacher in the local schools.

12.  $B = 1000(Bk - 0.63)^2$  where  $Bk$  is the proportion of [people of African American descent] by town):

- $B$  represents a calculated feature based on the proportion of people of African American descent in the town. It's derived from the formula provided.

13. **LSTAT (Percentage of lower status of the population):**

- LSTAT represents the percentage of the population classified as lower status. It provides insight into the socioeconomic status of the neighborhood.

These features collectively provide various aspects of the socioeconomic, environmental, and infrastructural characteristics of different neighborhoods in Boston. They serve as inputs to predictive models aiming to estimate housing prices based on these attributes.

< 2 / 2 > ⌂ ⌃ ⌄

Message ChatGPT...



ChatGPT can make mistakes. Consider checking important information.

