

**M.Sc. DATA ANALYTICS 2018-2019**

**MACHINE LEARNING**

**CA - 2**

**A large brick building

Description automatically generated**

**Boston Housing Price Prediction**

**using**

**Neural Network**

**In**

**Python**

**Using**

**tensorflow**

**BY:**

**Poonam Dhoot – 10399137@mydbs.ie**

**Sunmeet Thapar – 10506082@mydbs.ie**

**Machine Learning CA 2 – Python using TensorFlow**

**Prepared by:**

|  |  |
| --- | --- |
| Student Name(s) | Student ID number (email) |
| 1. Poonam Dhoot | 10399137@mydbs.ie |
| 1. Sunmeet Thapar | 10506082@mydbs.ie |

**Topic: Artificial Neural Network(ANN)**

Artificial neural networks are one of the main tools used in machine learning. As the “neural” part of their name suggests, they are brain-inspired systems which are intended to replicate the way that we humans learn. Neural networks consist of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. They are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract and teach the machine to recognize.

While neural networks (also called “perceptrons”) have been around since the 1940s, it is only in the last several decades where they have become a major part of artificial intelligence. This is due to the arrival of a technique called “backpropagation,” which allows networks to adjust their hidden layers of neurons in situations where the outcome doesn’t match what the creator is hoping for — like a network designed to recognize dogs, which misidentifies a cat, for example.

Another important advance has been the arrival of deep learning neural networks, in which different layers of a multilayer network extract different features until it can recognize what it is looking for.

**Brief Explanation of the dataset:**

1. Boston Housing Price Prediction – It is a very famous dataset available in package **sklearn**
2. In this project, we will evaluate the performance and predictive power of a model that has been trained and tested on data collected from homes in suburbs of Boston, Massachusetts. A model trained on this data that is seen as a good fit could then be used to make certain predictions about a home — in particular, its monetary value. This model would prove to be invaluable for someone like a real estate agent who could make use of such information on a daily basis.
3. It is Regression type of prediction. Since the values are numerical.

**Python Code:**

*# Import dependencies*

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

**from** **sklearn.datasets** **import** load\_boston

**import** **tensorflow** **as** **tf**

**import** **matplotlib.pyplot** **as** **plt**

%matplotlib inline

**import** **random**

*#Load Dataset*

boston = load\_boston()

*# Seperate Data into Features and Labels and load them as a Pandas Dataframe*

features\_df = pd.DataFrame(np.array(boston.data), columns=[boston.feature\_names])

features\_df.head()

labels\_df = pd.DataFrame(np.array(boston.target), columns=['labels'])

labels\_df.head()

combined\_data = pd.concat([features\_df,labels\_df], axis=1)

combined\_data.head()

*# Split data*

**from** **sklearn.model\_selection** **import** train\_test\_split

random.seed( 1000 )

X\_train, X\_test, y\_train, y\_test = train\_test\_split(features\_df, labels\_df, test\_size=0.2)

*# Scale Train data*

**from** **sklearn.preprocessing** **import** StandardScaler

scaler = StandardScaler()

scaler.fit(X\_train)

X\_train = pd.DataFrame(data=scaler.transform(X\_train), columns=X\_train.columns, index=X\_train.index)

X\_train = np.array(X\_train)

y\_train = np.array(y\_train)

type(X\_train), type(y\_train)

*# Scale Test data*

scal = StandardScaler()

scal.fit(X\_test)

X\_test = pd.DataFrame(data=scal.transform(X\_test), columns=X\_test.columns, index=X\_test.index)

X\_test = np.array(X\_test)

y\_test = np.array(y\_test)

type(X\_test), type(y\_test)

*# Define Feature columns*

features\_df.columns

*# Make Feature Columns*

feat\_cols = [tf.feature\_column.numeric\_column('x', shape=np.array(X\_train).shape[1:])]

*# Define Input Fuction*

input\_func = tf.estimator.inputs.numpy\_input\_fn({'x':X\_train}, y\_train, batch\_size=1, num\_epochs=2000, shuffle=**True**)

*# Build the model*

dnn\_model = tf.estimator.DNNRegressor(hidden\_units=[5,3],feature\_columns=feat\_cols, optimizer='Adam')

dnn\_model.train(input\_fn=input\_func, steps=2000)

*# Evaluate the model*

dnn\_model.evaluate(input\_fn=eval\_input\_func)

predictions = dnn\_model.predict(input\_fn=eval\_input\_func)

pred = list(predictions)

predicted\_vals = []

**for** pred **in** dnn\_model.predict(input\_fn=eval\_input\_func):

predicted\_vals.append(pred['predictions'])

print(predicted\_vals)

*# Performance Evaluation*

**from** **sklearn.metrics** **import** mean\_squared\_error

*# Calculate Mean Squared Error*

mse = mean\_squared\_error(predicted\_vals, y\_test)

print('Mean Squared Error [DNNRegrssor]: ',mse)

*# Improve the Performance by changing Parameters*

dnn\_model\_imp = tf.estimator.DNNRegressor(hidden\_units=[10,5,3],feature\_columns=feat\_cols, optimizer=tf.train.ProximalAdagradOptimizer(

learning\_rate=0.1,

l1\_regularization\_strength=0.001

))

dnn\_model\_imp.train(input\_fn=input\_func, steps=2000)

dnn\_model\_imp.evaluate(input\_fn=eval\_input\_func)

new\_predictions = dnn\_model\_imp.predict(input\_fn=eval\_input\_func)

new\_pred = list(new\_predictions)

new\_predicted\_vals = []

**for** new\_pred **in** dnn\_model\_imp.predict(input\_fn=eval\_input\_func):

new\_predicted\_vals.append(new\_pred['predictions'])

print(new\_predicted\_vals)

new\_mse = mean\_squared\_error(new\_predicted\_vals, y\_test)

print('Improved Mean Squared Error [DNNRegrssor]: ',new\_mse)

*# Compare Performance*

print('Old Mean Squared Error: ',mse)

print('New Mean Squared Error: ',new\_mse)

**Attachments:**

Boston Housing using Neural Network

1. Python code file (.ipynb)
2. Python code in Notepad
3. PowerPoint Presentation