Neural Networks vs. Machine Learning Algorithms.

HASSAN MOHAMAD
Hassanmohamad241@gmail.com
5716

LEBANESE UNIVERSITY - FACULTY OF ENGINEERING BRANCH III

Prepared for Dr. Youssef HARKOUS

January 31, 2022

Contents

List of Figures

Acknowledgments

First and foremost, we would like to thank the dean, the faculty director, and the head of the electrical and telecommunication department and of course Dr. Y. Harkous for whom we are presenting our work.

Abstract

This paper explains the meaning behind the terms artificial intelligence, machine learning, and deep learning, and compares the performance and results of two algorithms (Multiple linear regression and Keras Regression) on predicting house prices.

1 Introduction:

“Artificial Intelligence, deep learning, machine learning — whatever you’re doing if you don’t understand it — learn it. Because otherwise, you’re going to be a dinosaur within 3 years.”-Mark Cuban-
Among all living creatures on this planet, humans are distinguished by being the smartest and most brilliant –at least that’s what they say-. It all came down to the fact that the human brain is one of the most complex and efficiently built organs in the universe, shaped and evolved over more than 1.7 million years. But how can humans think? What exactly happens when we try to learn, solve, think, etc…During the last century, with the creation of computers, we started approaching these questions faster than ever. We created algorithms that enabled computers to solve problems that we could never solve before, even though we were the inventors of these algorithms. And every day we approach the invention of an AI that can pass “The Turing test”, which is a test of a machine's ability to exhibit intelligent behavior equivalent to, or indistinguishable from, that of a human. It is just a matter of time before we reach the point of technological singularity.
We may or may not be able to define human intelligence, but here we are only interested in answering the following questions, what is artificial intelligence (AI). What does the term “Machine learning’’ (ML) mean? And what are neural networks (Deep learning)? In this project, we will answer these questions, discuss the simplest machine learning algorithms, with small neural networks, and compare their performance on "House price prediction".

2 Materials used:

In this project, we’re going to create a model to predict House prices based on various factors across different markets using the Python programming language alongside with its different

modules and libraries like: Numpy, Panda, Matplotlib, Seaborn, TensorFlow, Keras. Of course we can't study the performance of the models without having a data set in our hands, so we will be using a data set called: "Kaggle-kc house" dataset with the following features: Date sold, Price, number of bedrooms, number of bathrooms, square footage of the home, square footage of the lot, number of floors, waterfront, number of views, condition, grade, square footage of house apart from basement, square footage of the basement, year built, year renovated, zip code, latitude coordinate, longitude coordinate, Living room area in 2015, lotSize area in 2015.

	0	1	2	3	4
id	7129300520	6414100192	5631500400	2487200875	1954400510
date	10/13/2014	12/9/2014	2/25/2015	12/9/2014	2/18/2015
price	221900	538000	180000	604000	510000
bedrooms	3	3	2	4	3
bathrooms	1	2.25	1	3	2
sqft_living	1180	2570	770	1960	1680
sqft_lot	5650	7242	10000	5000	8080
floors	1	2	1	1	1
waterfront	0	0	0	0	0
view	0	0	0	0	0
condition	3	3	3	5	3
grade	7	7	6	7	8
sqft_above	1180	2170	770	1050	1680
sqft_basement	0	400	0	910	0
yr_built	1955	1951	1933	1965	1987
yr_renovated	0	1991	0	0	0
zipcode	98178	98125	98028	98136	98074
lat	47.5112	47.721	47.7379	47.5208	47.6168
long	-122.257	-122.319	-122.233	-122.393	-122.045
sqft_living15	1340	1690	2720	1360	1800
sqft_lot15	5650	7639	8062	5000	7503

Figure 1: First 5 records of our dataset.

3 Definition of different terms:

Artificial intelligence is a poorly defined term, which contributes to the confusion between it and machine learning. Artificial intelligence is essentially a system that seems smart. That's not a very good definition, though, because it's like saying that something is 'healthy'. These behaviors include problem-solving, learning, and planning, for example, which are achieved through analyzing data and identifying patterns within it in order to replicate those behaviors. Machine learning, on the other hand, is a type of artificial intelligence, where artificial intelligence is the overall appearance of being smart, machine learning is where machines are taking in data and learning things about the world that would be difficult for humans to do. Artificial intelligence is a technology that enables a machine to simulate human behavior. Machine learning is a subset of AI which allows a machine to automatically learn from past data without programming explicitly. The goal of AI is to make a smart computer system like humans to solve complex problems. The goal of ML is to allow machines to learn from data so that they can give accurate output. Machine learning algorithms are traditionally divided into three broad categories, let's discuss the most two important ones:

3.1 Supervised learning:

Supervised learning algorithms build a mathematical model of a set of data that contains both the inputs and the desired outputs. The data is known as training data, and consists of a set of training examples. Each training example has one or more inputs and the desired output.

If you're learning a task under supervision, someone is present, judging whether you're getting the right answer. Supervised learning is very similar to that. Types of supervised learning algorithms include classification and regression that we will discuss in the next sections.

3.2 Unsupervised learning:

Unsupervised learning algorithms take a set of data that contains only inputs, and find structure in the data, like grouping or clustering of data points. The algorithms, therefore, learn from test data that has not been labeled, classified or categorized. Instead of responding to feedback, unsupervised learning algorithms identify commonalities in the data and react based on the presence or absence of such commonalities in each new piece of data. Unsupervised learning allows users to perform more complicated tasks compared to supervised learning. A couple of algorithms are used in unsupervised learning, such as clustering, and probabilistic decision. In clustering, k means clustering is one of the methods popularly used in unsupervised machine learning.

4 Algorithms in machine learning:

In this section we will discuss different machine learning algorithms that are categorized under the supervised learning (Regression) or the unsupervised learning.

4.1 Linear regression:

The purpose of a regression model is to understand the relationship between features and target.

Given a data $\{y_i, x_{i1}, \ldots, x_{ip}\}_{i=1}^{n}$ set of *n* statistical units, a linear regression model Assumes that the relationship between the dependent variable *y* and the *p*-vector of regressors **x** is linear:

$$y = \alpha + \beta x$$

By expanding to get a quadratic expression in α and β we can derive values of $\widehat{\alpha}$ and $\widehat{\beta}$ estimators of α and β respectively:

$$\widehat{\alpha} = \bar{y} - (\widehat{\beta}\,\bar{x}),$$

$$\widehat{\beta} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$\bar{x}$ and $\bar{y}$ are the average of the X_i and Y_i, respectively.

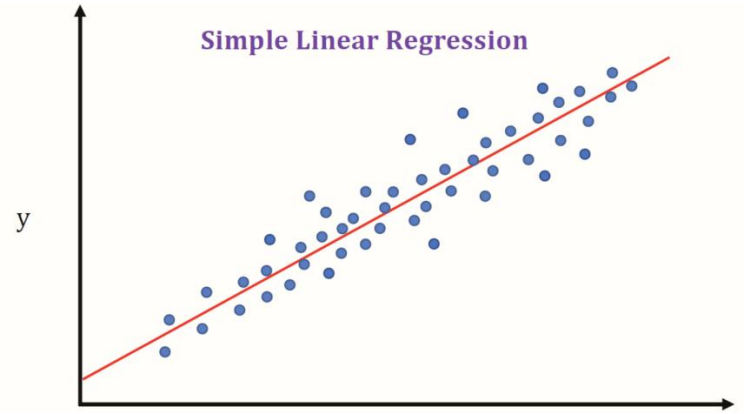


Figure 2: Graph representing a simple linear regression model.

4.2 Multiple regression:

In our project we used **Multiple Linear Regression** which is an extension of Simple Linear Regression and assume that there is a linear relationship between a dependent variable Y and independent variable**s** X.

Formula and Calculation of Multiple Linear Regression:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon$$

Where:

y_i = Dependent variable.

x_i = Explanatory variables.

β_0 = Y-intercept variables.

β_p = Slope coefficients for each explanatory variable.

ϵ = The model's error term (also known as the residuals).

The multiple regression model is based on the following assumptions:

1. There is a linear relationship between the dependent variables and the independent variables.
2. The independent variables are not too highly correlated with each other.
3. y_i Observations are selected independently and randomly from the population.
4. Residuals should be normally distributed with a mean of 0 and variance σ.

4.3 Classification and nearest neighbor:

Classification is a process of categorizing a given set of data into classes, it can be performed on both structured and unstructured data. The process starts with predicting the class of given data points. The classes are often referred to as target, label or categories.

There are countless of algorithms used to perform classification, some of which are supervised, and others are unsupervised (Naïve Bayes, Decision tree, K-nearest neighbors). With K-nearest neighbors being the most popular and known, we will focus only on this algorithm.

It is actually called K-Nearest-Neighbor (K-NN for short), where K being the number of the nearest neighbors of a vector. K-NN is one of the most basic algorithms used in classification problems, the idea behind it is to find the K nearest neighbors of a given instance from a whole data set so that this instance would belong to the class that has the most neighbors, for example if I am classifying an instance x with K=3 and we found those 3 nearest neighbors y1, y2, y3, if y1 and y3 belong to class c0 and y2 belongs to class c1 then x would be classified just like y1 and y3 (class c0). But how do we find the nearest neighbor? Based on what? That is where the math starts, to find the nearest neighbor we just have to calculate the distance of the vector x to all the instances in the data set and x's nearest neighbor would be the instance with the smallest distance from it, there are many methods to calculate the distance between two vectors but we the most widely used distance is the Euclidean distance:

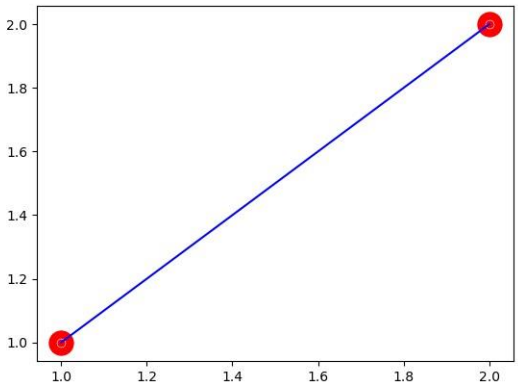


Figure 3: Euclidean Distance between two points in a two dimensional plane.

$$ED(x,y) = \sqrt{(x_1 - y_1)^2 + (x_2 - y_2)^2}$$

This is if *x* and *y* are two points in a two dimensional plane, this how it's calculated if x and y are n dimensional vectors:

$$ED(x,y) = \sqrt{\sum_{i=1}^{i=n}(x_i - y_i)^2}$$

5 Neural Networks (deep learning):

To define neural networks we have to know why they were called like that. Neural networks are a set of algorithms inspired by the functionality of the human brain, for example when we hear a sound, these sound waves are converted to brain waves signals to be processed but what we call the neurons in our brain, similarly this is the idea behind neural networks but of course we cannot say that the algorithms we build are an explanation of how our brain

studies information and learn new things, so we only took the nomenclature and called it neural networks and create artificial neuron networks (ANN's) to do the work.

So after this explanation above you would ask yourself, what do we mean by artificial neural networks? We can consider ANN's like a human brain, it needs to learn patterns from a given data, analyze the data, know how to classify the instance of a data, that data can be images, sounds etc... To go even deeper, ANN's are not more than learning an approximation of a function $y = f(x)$, now what do we mean by approximation of a function? To answer this question let's start by the architecture of a neural network first, neural networks are made of layers, each layer has a functionality different than the others and a specific number of artificial neurons:

5.1 Artificial Neurons:

An artificial neuron can be defined as an input output system, where x is the input and y is the output, with a very simple linear relation between y and x:

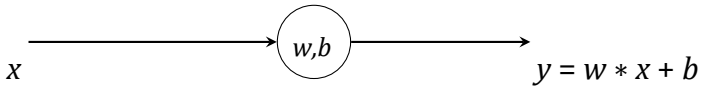


Figure 4: Artificial Neuron.

Where w is called the weight and b the bias. The idea is that we have an input x and an output y, we want to find the linear relation between x and y so eventually find the correct w and b. How to do this? ANN's most important concepts are forward and backward propagation, in the forward propagation phase our goal is to calculate the output $y^0 = w * x + +b$, backward propagation consists on calculating the loss between the real target y and the predicted target y^0 and adjust the weight w and bias b and get closer to the correct values. How to adjust w and b and how to know when to stop? Our goal in ANN's is to minimize the loss as between y^0 and y as much as possible and each time we calculate the loss (which should be decreasing from what we just explained) we adjust the weight and bias using a specific optimizer. Many optimizers are used these days, we are going to talk about a simple one called Gradient descent.

5.2 Layers:

There is a lot of layers that we can explain in deep learning but we're going to explain its meaning in general. A layer in ANN is a container of neurons, it holds a collection of neurons

that performs the same function and the same activation function to provide a specific output.

5.2.1 Input layer:

Every neural network must have an input layer, its job is to receive the input data from web service or a CSV file etc...

5.2.2 Output layer:

Also every neural network must have an output layer, it is responsible for producing the final results of the network, for example for a digital recognition NN, the output layer consists of ten neuron labeled from 0 to 9.

5.2.3 Hidden layer:

Hidden layers reside in between the input and output layers, they are not visible to the external system, there could be 0 or more hidden layers in an ANN. The number of neurons in each hidden layer is dependent and changed based on the project.

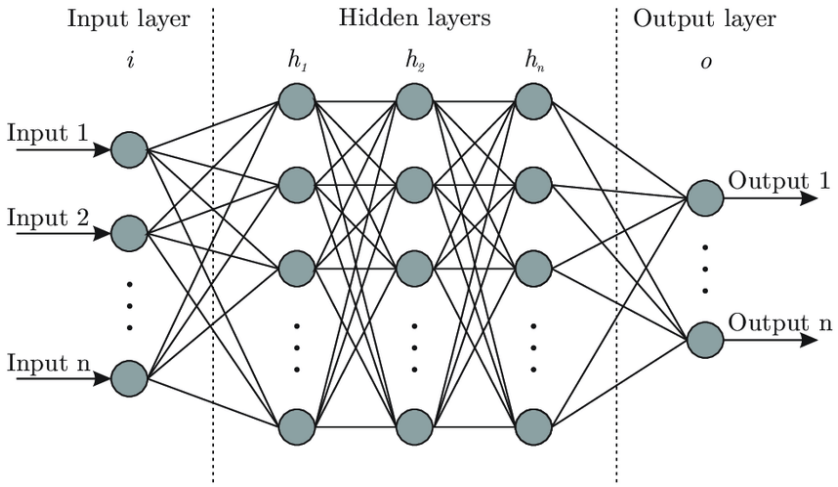


Figure 5: Example of an Artificial Neural Network.

5.3 Gradient Descent Optimizer:

We explained before that after we calculate the loss we have to adjust the weight w and bias b accordingly to minimize a certain "cost function" that we previously define, which represents how far the network's output is relative to the desired output.

For the sake of simplicity, let's assume one neuron in each layer, and define the following terms:

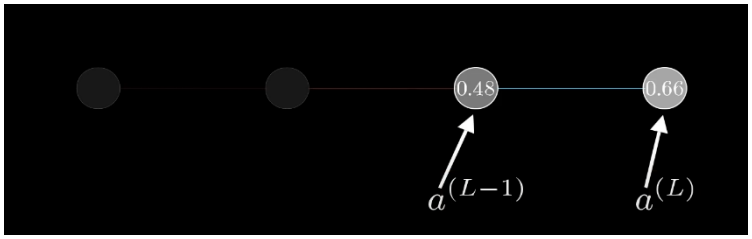


Figure 6: Simple neural network with 1 neuron in the all the layers.

- $a^{(L)}$: Activation of the last neuron.
- $a^{(L-1)}$: Activation of the previous neuron.
- y : Desired value.
- Cost function: $C_0(\dots) = \left(a^{(L)} - y\right)^2$.
- Weight: $w^{(L)}$.
- Bias: $b^{(L)}$.

Let's also define z (L): $z^{(L)} = w^{(L)}a^{(L-1)} + b^{(L)}$.

Then finally $a^{(L)} = \sigma\left(z^{(L)}\right)$ where σ is the sigmoid function. (We will explain the role of sigmoid function in the next section).

In the gradient descent algorithm, our aim is to minimize the cost function ${C_0}^{(L)}$ by changing the weight $w^{(L)}$ and the bias $b^{(L)}$.

To do so, we take the partial derivative of ${C_0}^{(L)}$ with respect to $w^{(L)}$ first, then $b^{(L)}$. (We're going to focus on $w^{(L)}$)

$\frac{\partial {C_0}^{(L)}}{\partial w^{(L)}} = \frac{\partial z^{(L)}}{\partial w^{(L)}} \cdot \frac{\partial a^{(L)}}{\partial z^{(L)}} \cdot \frac{\partial {C_0}^{(L)}}{\partial a^{(L)}} = 2\left(a^{(L)} - y\right).\ \sigma'\left(z^{(L)}\right).\ a^{(L-1)}.$

Now that we have found this expression for a specific training example, we find the average across all training examples:

$$\frac{\partial C}{\partial w^{(L)}} = \frac{1}{n} \sum_{k=0}^{n-1} \frac{\partial C_k}{\partial w^{(L)}}$$

And that is just one component of the gradient vector ∇C which is then built up using partial derivatives of the cost function, with respect to all these weights and biases.

Now we keep iterating the chain rule backwards to see how sensitive the cost function is to previous weights and biases.

If we step back and look to the whole picture we can simply say that we adjust the weight and bias in this manner:

$$w = w - \alpha * \frac{\partial L(w, b)}{\partial w}$$

And

$$b = b - \alpha * \frac{\partial L(w, b)}{\partial b}$$

Where α is what we call the learning rate which is a hyper parameter, we mean by hyper-parameter that it doesn't have a rule to be given a value because it depends on the data set and the functionality of the layer and the number of neurons etc...

So the idea is we start with random values given for the weights and the biases, we calculate the loss function, adjust w and b using the gradient descent explained above then keep doing the same work e times, where e is called the number epochs and in each epoch we perform the forward and backward propagation.

5.4 Activation Functions:

Till now we worked only with linearity between input and output, but what if it doesn't exist? Shouldn't we introduce some non-linearity to out neural network? Using what we call activation functions we can solve this problem to make our output limited to a certain region

in the real numbers line. For example if we want the neurons to contain only number between 0 and 1, we use the Sigmoid function. This function crunches the number line to the region]0,1[.

The activation function is also a hyper parameter which means there isn't a rule to choose one, several may be used like:

5.4.1 Relu:

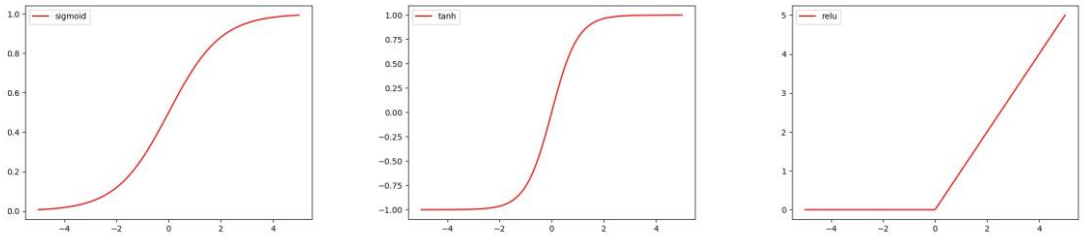
$$f(x) = max(0,x)$$

5.4.2 Hyperbolic tangent:

$$f(x) = \tanh(x) = \frac{2}{1 + e^{-2x}} - 1$$

5.4.3 Sigmoid:

$$f(x) = \frac{1}{1 + e^{-x}}$$



(a) Sigmoid. (b) Tanh. (c) Relu.

Figure 7: Examples of activation functions.

5.5 Adam Optimization Algorithm:

In the section **5.3**, we discovered the role of an optimizer in an artificial neural network, and we explained the most known and widely used optimizer –Gradient descent-, but in our project, we used a different optimizing algorithm called "Adam" (the name Adam is derived

from adaptive moment estimation). Unfortunately, to explain how Adam algorithm works, we need to explain two other optimizing algorithms, "Adaptive Gradient Algorithm", and "Root Mean Square Propagation" which will take from us a lot of more pages and effort to just explain an optimizing algorithm, but since we explained the general principal behind optimizing algorithms in **5.3**, we will put more details on "Adam" in the appendix part.

6 Putting It All Together:

You can find the code of the project Here.

6.1 Dealing with the data:

The data in our data set pass through the following sequence of actions:

- Read the data from the csv file.
- Classify the data into integers, floats, and objects (only the data parameter).
- Drop the unnecessary features that doesn't affect the prediction (ID, Zip code).
- Check for Null values (as these values can cause unexpected results after applying the algorithms).
- Some steps include visualizing some data to understand it more, and break it down to graph it, and get rid of outliers.

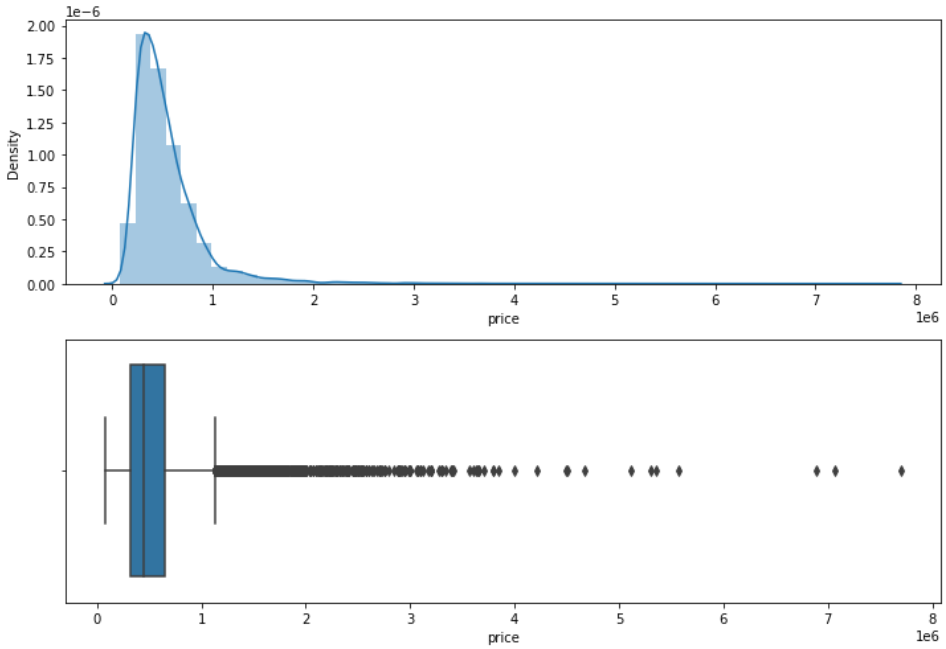


Figure 8: distribution plot of price.

We can notice that most of the prices are between 0 and around 1M with few outliers close to 8 million (fancy houses). It would make sense to drop those outliers in our analysis.

- Split the data into "Train data" for the training part and "Test data" for the testing at the end.
- Features (X): The columns that are inserted into our model will be used to make predictions.
- Prediction (y): Target variable that will be predicted by the features
- Scale the data as it will help us see all the variables from the same lens (same scale), it will also help our models learn faster.

6.2 Implantation of the Artificial Neural network:

Since we have 19 features, let's insert 19 neurons as a start, 3 hidden layers and 1 output layer due to predict house Price. (Each hidden layer contains 19 neurons as well)

Then, we train the model for 400 epochs, (in each epoch we perform the forward and backward propagation) and each time record the training and validation accuracy in the history object. To keep track of how well the model is performing for each epoch, the model will run in both train and test data along with calculating the loss function.

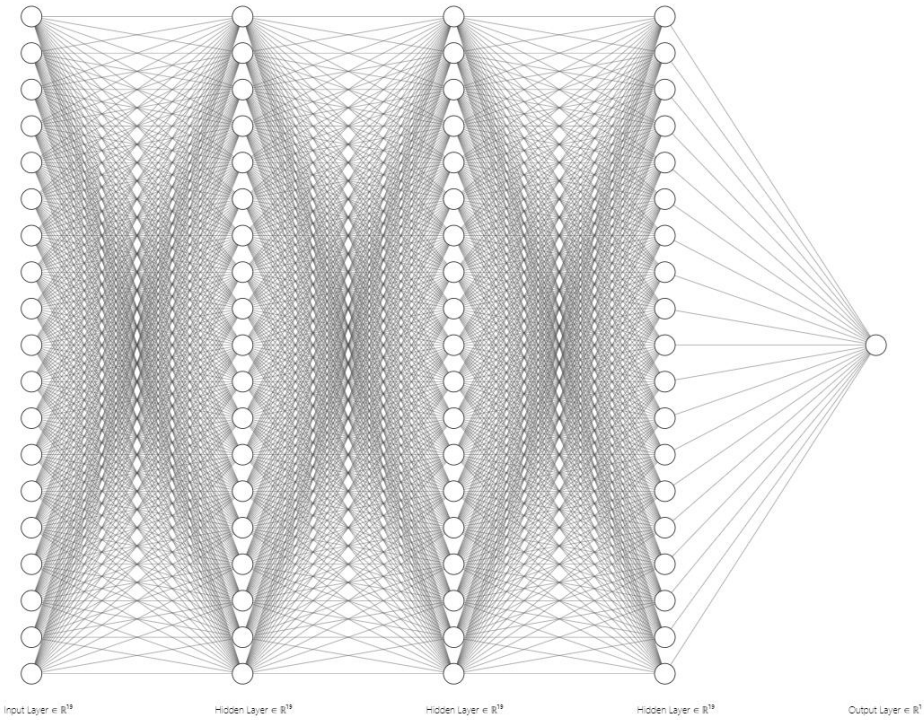


Figure 9: Visual Representation of our Neural Network.
https://alexlenail.me/NN-SVG/

7 Results:

Let's compare actual output and predicted value to measure how far our predictions are from the real house prices.

Mean Absolute Error: MAE= $\frac{\sum_{i=0}^{n}|y_i-x_i|}{n}$ (y_i: predicted values / n: number of data points)

Mean Squared Error: MSE= $\frac{1}{n}\sum_{i=1}^{n}(Y_i-\hat{Y}_i)^2$ ($\hat{Y}_i$: $predicted\ values$)

Root Mean Squared Error: RMSE= $\sqrt{\frac{\sum_{i=0}^{N}(x_i-\hat{x}_i)^2}{N}}$ (N: number of non-missing data points)

Variance score is: $1-\frac{Var(y-\hat{y})}{Var(y)}$

7.1 Method 1: Multiple linear regression:

Mean Absolute Error: 125933.74
Mean Squared Error: 40601153229.62
Root Mean Squared Error: 201497.28
Variance score is: 0.714

	ACTUAL	PREDICTED
0	349950.0	530667.784210
1	450000.0	667025.076946
2	635000.0	553195.043391
3	355500.0	346657.166101
4	246950.0	61378.186019
5	406550.0	481162.809294
6	350000.0	312819.788488
7	226500.0	273833.027682
8	265000.0	280571.649291
9	656000.0	532966.844438

Figure 10: Multiple Linear Regression Results

Results show that using this method we are off about 20% (comparing mean absolute error and mean of price).

7.2 Method 2: Keras Regression (Artificial Neural Network):

Mean Absolute Error: 100428.63
Mean Squared Error: 26227797826.74
Root Mean Squared Error: 161949.99
Variance score is: 0.81

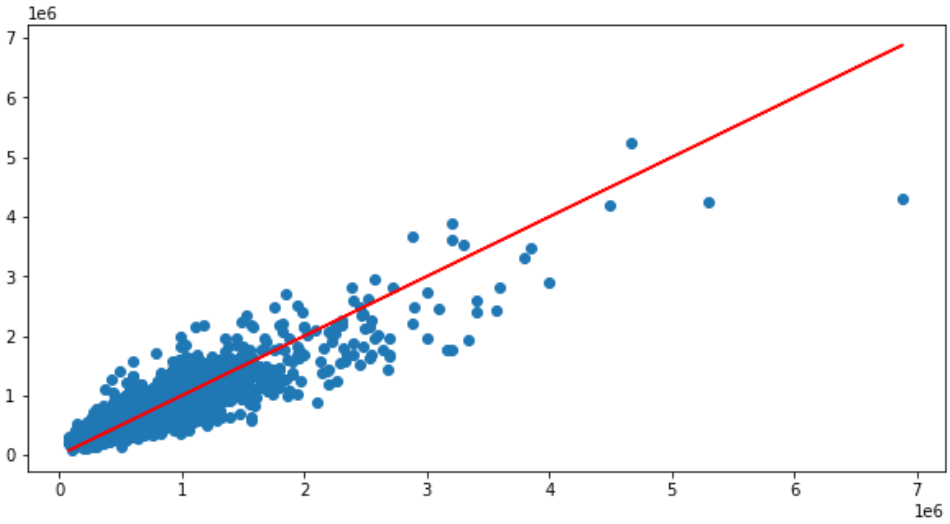


Figure 11: Keras regression results.

Results also show that we are off about 20% (comparing mean absolute error and mean of price).

8 Discussion:

	Multiple linear regression	Keras regression
Mean Absolute Error	125933.74	100428.63
Mean Squared Error	40601153229.62	26227797826.74
Root Mean Squared Error	201497.28	161949.99
Variance score (%)	71.40	80.88

The score of our Multiple Linear Regression is around 70%, so this model had room for improvement. Then we got an accuracy of ~81% with Keras Regression model.

Also, notice that RMSE (loss function) is lower for Keras Regression model which shows that our prediction is closer to actual rating price.

9 Conclusion:

In this small paper, we briefly explained the meaning behind the terms of AI and Machine learning, to finally transmit to the definition of a neural network. Neural Networks are essentially a part of Deep Learning, which in turn is a subset of Machine Learning. So, Neural Networks are nothing but a highly advanced application of Machine Learning. While a Machine Learning model makes decisions according to what it has learned from the data, a Neural Network arranges algorithms in a fashion that it can make accurate decisions by itself. Thus, although Machine Learning models can learn from data, in the initial stages, they may require some human intervention, and that is what gives neural networks the upper hand, they are more free and capable of learning nearly anything without the need of creating a very complicated algorithm or code.

As we saw the architecture behind neural networks is highly inspired by the structure of biological neurons. Artificial neural networks are probably our first attempt at trying to simulate a human mind, but we are still way far away. These neural networks use an incredible amount of power and take a lot of space just trying to simulate what a human brain can do consuming way less power and taking way smaller space, and still, our brains can outperform them in many tasks. Our paper showed one example of when neural networks perform better than regular machine learning algorithms, but there's still a huge problem that faces us while working on neural networks, "Computational power".

While performing the forward and backward propagations, our computers consume a lot of energy multiplying the corresponding matrices to converge to the right weights and biases, and that is a fundamental problem with our current digital computers. A recent video done by the famous Youtuber "Derek Muller" on his channel "Veritasium" showed that it is much more effective (energetically speaking) to use analog computers while dealing with neural networks instead of the typical digital ones.

All of the above forces us to question the future of AI, after beating us in Chess, Marketing, and many other domains, and replacing us in jobs that we thought we only could do, we wonder what's next? And as Elon Musk once said: *"AI will be the best or worst thing ever for humanity."*

10 References:

https://tslearn.readthedocs.io/en/stable/

https://www.investopedia.com/terms/m/mlr.asp

https://alexlenail.me/NN-SVG/

https://machinelearningmastery.com/linear-regression-for-machine-learning/

https://numpy.org/

https://matplotlib.org/

https://docs.python.org/3/library/tkinter.html

https://docs.python.org/3/

https://pandas.pydata.org/docs/

https://seaborn.pydata.org/

https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.html

https://keras.io/

https://www.upgrad.com/blog/machine-learning-vs-neural

https://www.analyticsinsight.net/the-difference-between-artificial-intelligence-and-machine-learning

https://www.upgrad.com/blog/machine-learning-vs-neural-networks

https://medium.com/geekculture/a-2021-guide-to-improving-cnns-optimizers-adam-vs-sgd-495848ac6008

https://www.pluralsight.com/guides/regression-keras

11 Appendices:

11.1 Appendix A: Diving into Adam Optimizing algorithm:

Adam optimizer is the extended version of stochastic gradient descent which could be implemented in various deep learning applications such as computer vision and natural language processing in the future years. Adam was first introduced in 2014. It was first presented at a famous conference for deep learning researchers called ICLR 2015. It is an optimization algorithm that can be an alternative for the stochastic gradient descent process. The name is derived from adaptive moment estimation. The optimizer is called Adam because uses estimations of the first and second moments of the gradient to adapt the learning rate for each weight of the neural network. The name of the optimizer is Adam; it is not an acronym. Adam is proposed as the most efficient stochastic optimization which only requires first-order gradients where memory requirement is too little. Before Adam, many adaptive optimization techniques were introduced such as AdaGrad, RMSP which have good performance over SGD but in some cases have some disadvantages such as generalizing performance which is worse than that of the SGD in some cases. So, Adam was introduced which is better in terms of generalizing performance. Also in Adam, the hyper parameters have intuitive interpretations and hence required less tuning. Adam performs well. But in some cases, researchers have observed Adam doesn't converge to the optimal solution, SGD optimizer does instead. In a diverse set of deep learning tasks sometimes Adam optimizers have low generalizing performance. According to the author Nitish Shirish Keskar and Richard Socher, switching to SGD in some cases show better generalizing performance than Adam alone.
Here is a link to a brief explanation of the theory behind Adam.

11.2 Appendix B: Analog vs. digital computers (dealing with ANNs)

Here is a link to Derek's video on Youtube.