House price prediction using Multiple Linear Regression and Keras Neural Network model

Arindam Saha, 002011701091, Production Engineering

Under the guidance of Professor Dr. Debamalya Banerjee

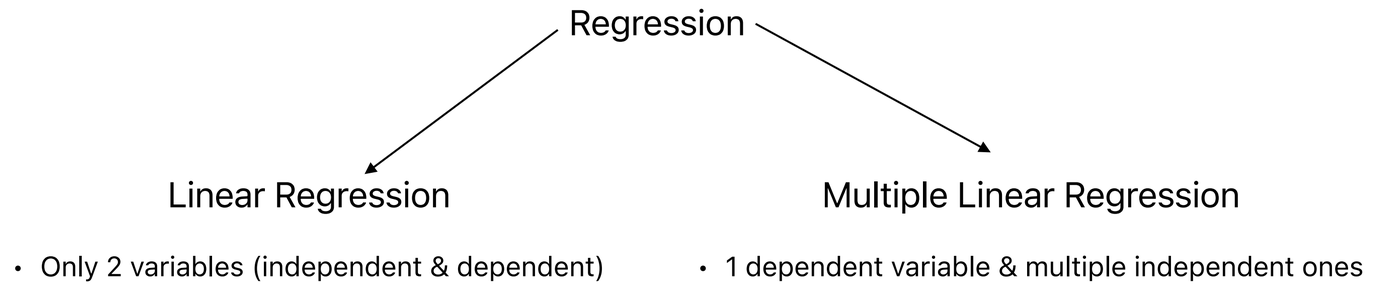
Contents

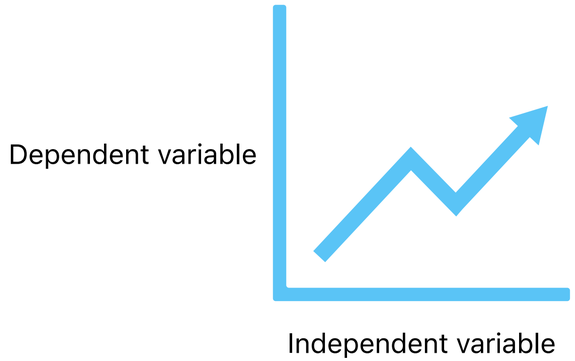
Advantages of House price prediction

- By having accurate predictions of house prices, individuals and organizations can make more informed decisions when buying or selling properties. This can lead to better investment decisions and increased profitability.
- Predictions of house prices can provide valuable insights into the real estate market, allowing individuals and organizations to better understand market trends and make more informed decisions.
- Predictions of house prices can save time by eliminating the need to manually analyze market data, allowing individuals and organizations to focus on other important tasks.
- Predictions of house prices can assist in forecasting future market trends, which can be used to make informed decisions about future investments.

Multiple Linear Regression

- Regression is the quantitative measure of relationship between dependent and independent variables.
-





Multiple Linear Regression is given by the following eq:

$$y = b0 + b1x1 + b2x2 + ... + bnxn - E$$

Assumptions

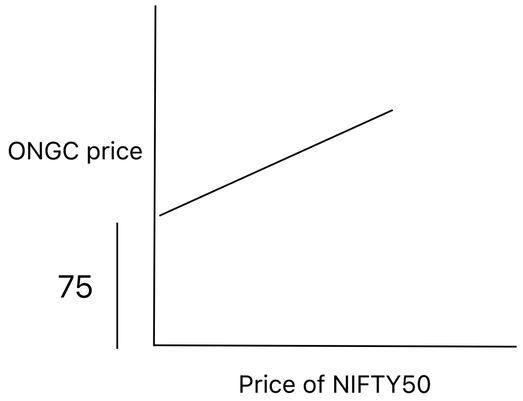
1. Linear relationship between two independent variables must be present
2. Independent variables must not be highly correlated to each other
3. y observations are selected randomly and independently from population
4. Residuals must be normally-distributed, with a mean of 0 and standard deviation of 1

For example, suppose the price of ONGC share price. It depends on multiple factors including the price of NIFTY50, price of oil, interest rate, price of future derivatives, stock price of other oil companies etc.

Suppose the equation can be given by the following -

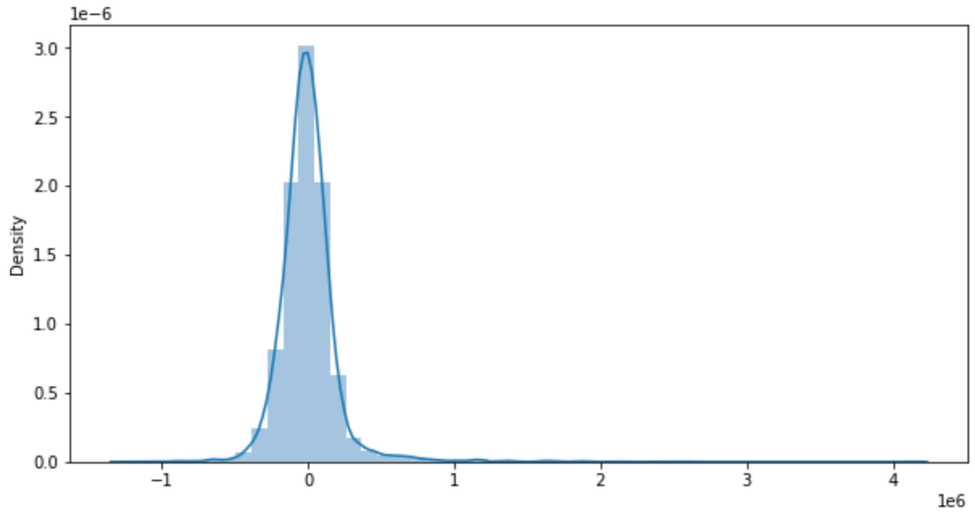
ONGC price = (75 - 1.5 * i + 7.8 * oil price + 3.2 * NIFTY50 + 5 * oil futures - E)

And R-sq = 83.4%



Implementation in Python

- Get independent (x) and dependent variable (y) from dataset
- Split train & test dataset
- Feature Scaling
- Do linear regression on x & y train splits
- Find intercept & slope
- Predict test result
- Put result in a data frame
- Check accuracy and errors



(y_test - y_pred)

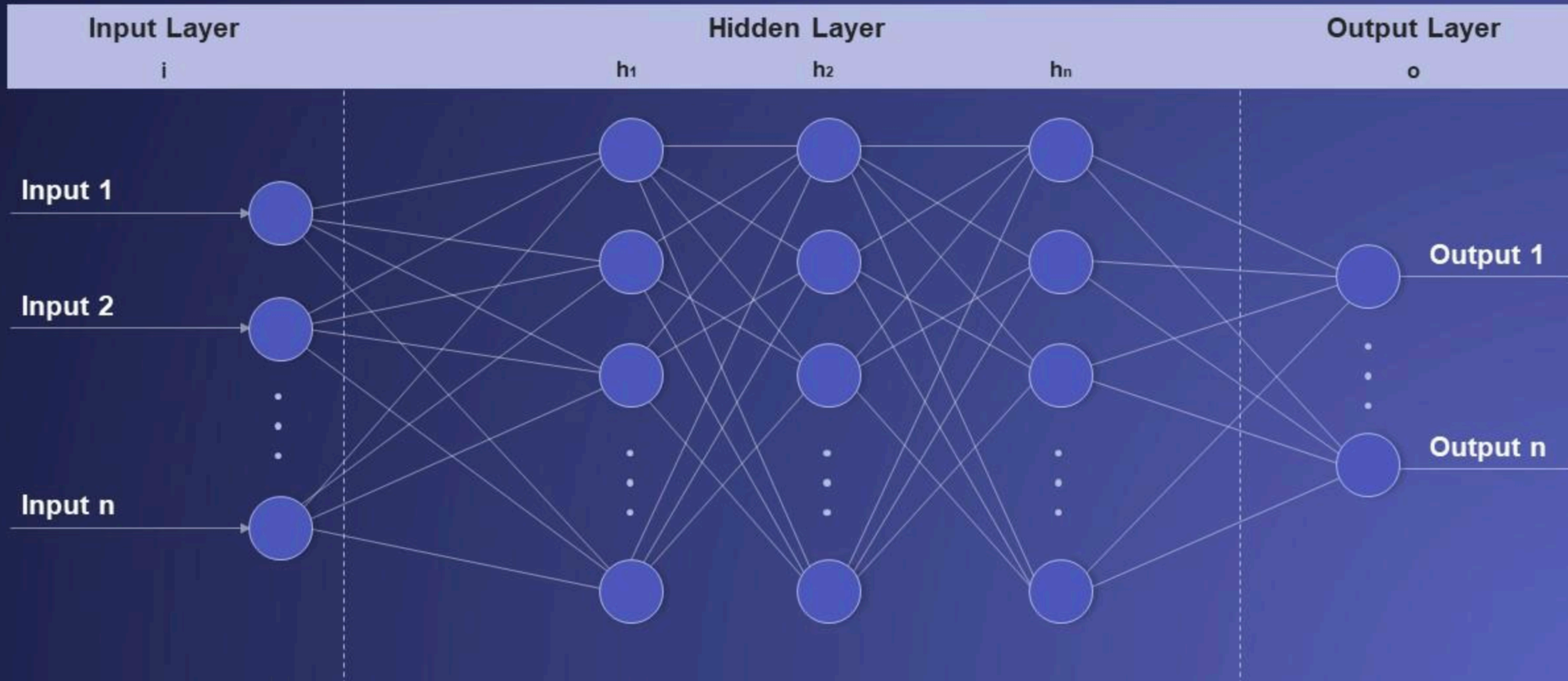
Price Prediction.py > ...

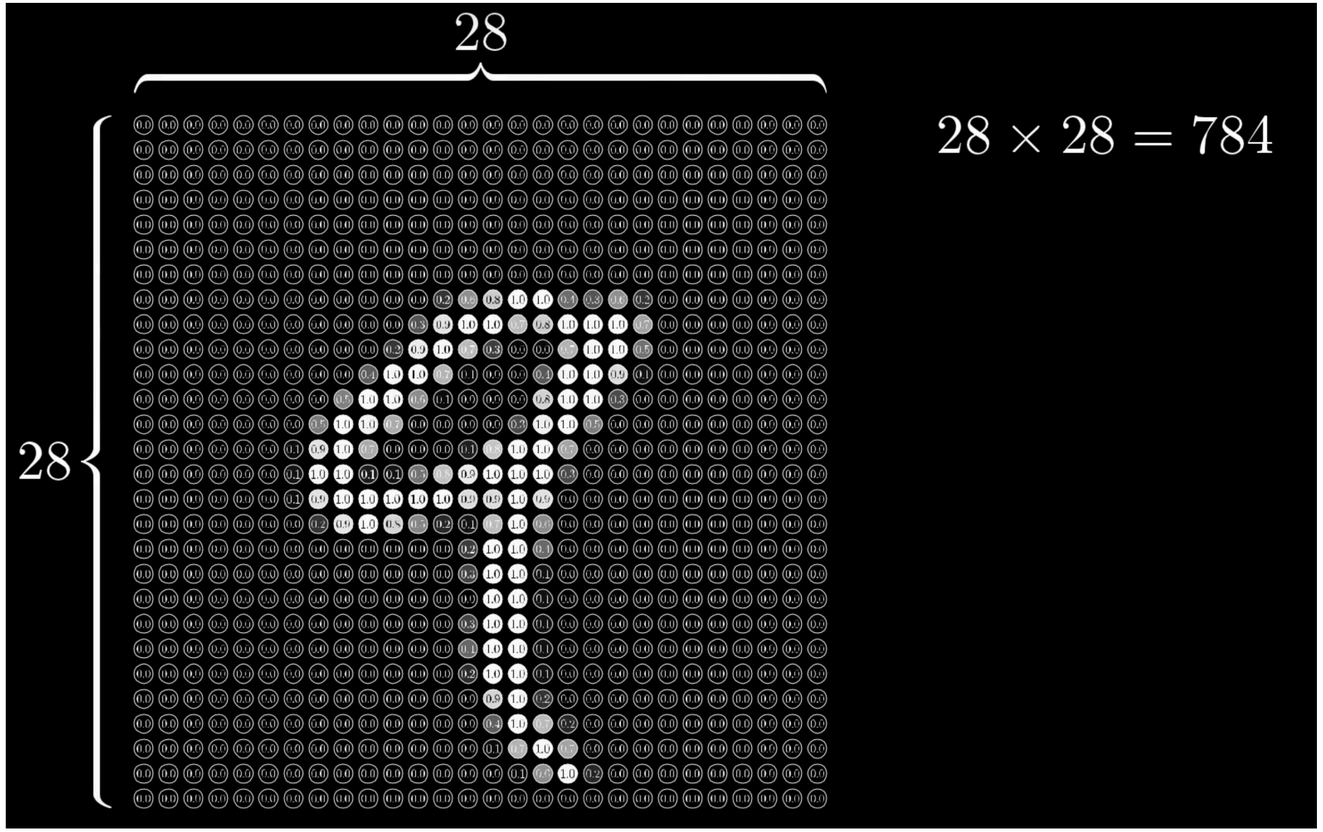
```
"""### Preparing Dataset (Scaling and Train-Test Split)"""

X = Data.drop('price',axis =1).values
y = Data['price'].values

#splitting Train and Test
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=101)

"""## Feature Scaling"""

#standardization scaler - fit&transform on train, fit only on test
from sklearn.preprocessing import StandardScaler
s_scaler = StandardScaler()
X_train = s_scaler.fit_transform(X_train.astype(np.float))
X_test = s_scaler.transform(X_test.astype(np.float))

"""# 1. Multiple Linear Regression"""

#Liner Regression
from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
regressor.fit(X_train, y_train)

#evaluate the model (intercept and slope)
regressor.intercept_
regressor.coef_

y_predd = regressor.predict(X_test)
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_predd})
df1 = df.head(10)
df1

fig = plt.figure(figsize=(10,5))
residuals = (y_test- y_pred)
sns.distplot(residuals)

from sklearn import metrics

print('Mean Absolute Error: {:.2f}'.format(metrics.mean_absolute_error(y_test, y_predd)))
print('Mean Squared Error:{:.2f}'.format(metrics.mean_squared_error(y_test, y_predd)))
print('Root Mean Squared Error:{:.2f}'.format(np.sqrt(metrics.mean_squared_error(y_test, y_predd))))
print('Variance score is: {:.2f}'.format(metrics.explained_variance_score(y_test,y_predd)))

# we are off about 20% (comparing mean absolut error and mean of price)
Data['price'].mean()
```

Neural Network using Keras

Architecture of artificial neural network

This slide demonstrates the architecture of Artificial Neural Network, which includes three layers such as input, hidden, and output.



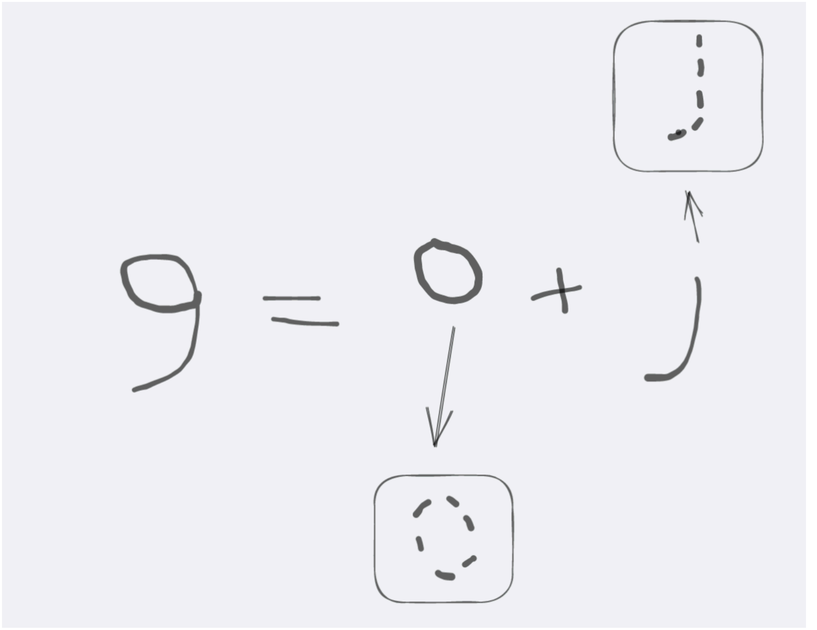


Neurones are mathematical function that collects
& classifies information according to special architecture

Each Neurone is a perceptron, and is similar to
Multiple linear regression

Text Recognition System

Recognition → ‘Re’ + ‘cog’ + ‘ni’ + ‘tion’

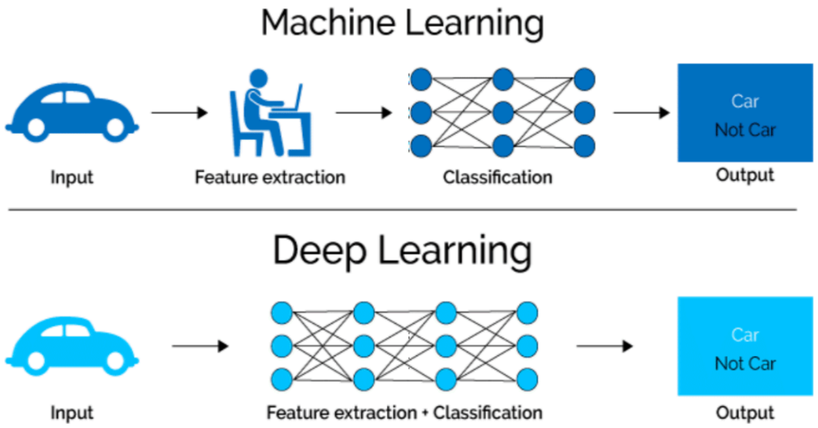


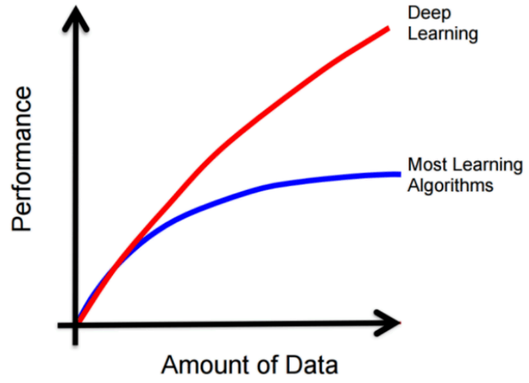
Types of Neural Networks

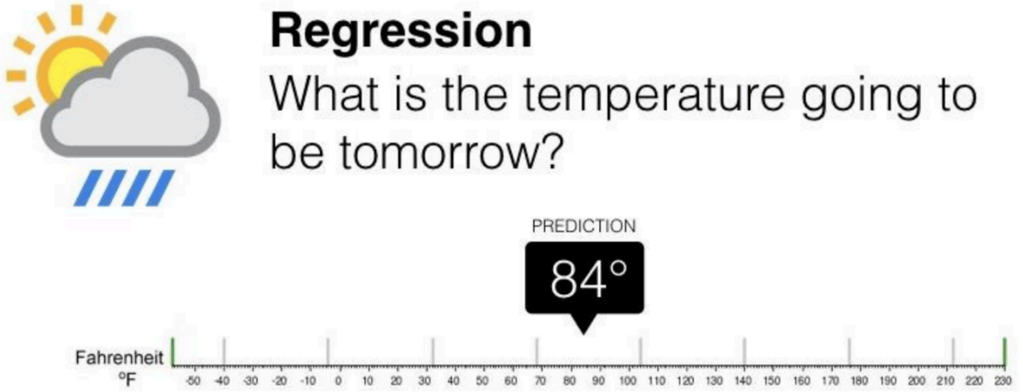
1. Feed-forward Neural Network
2. Recurrent Neural Network
3. Convolutional Neural Network
4. Deconvolutional Neural Network
5. Modular Neural Network

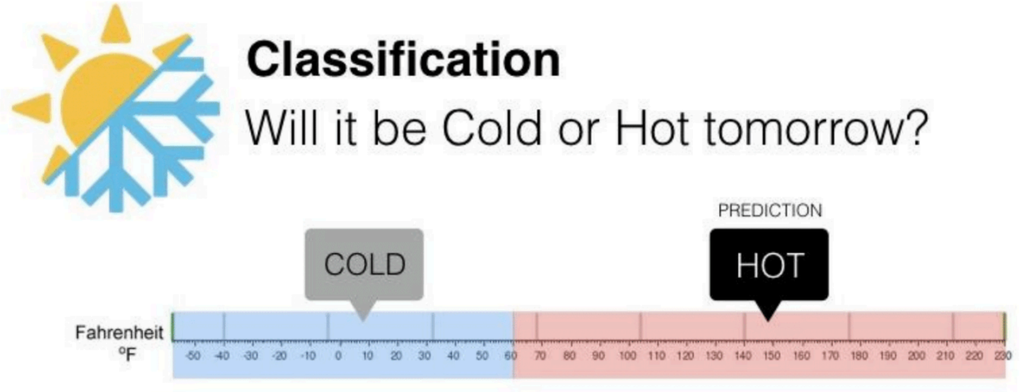
Deep Neural Network containing multiple layers, does its feature scaling itself.

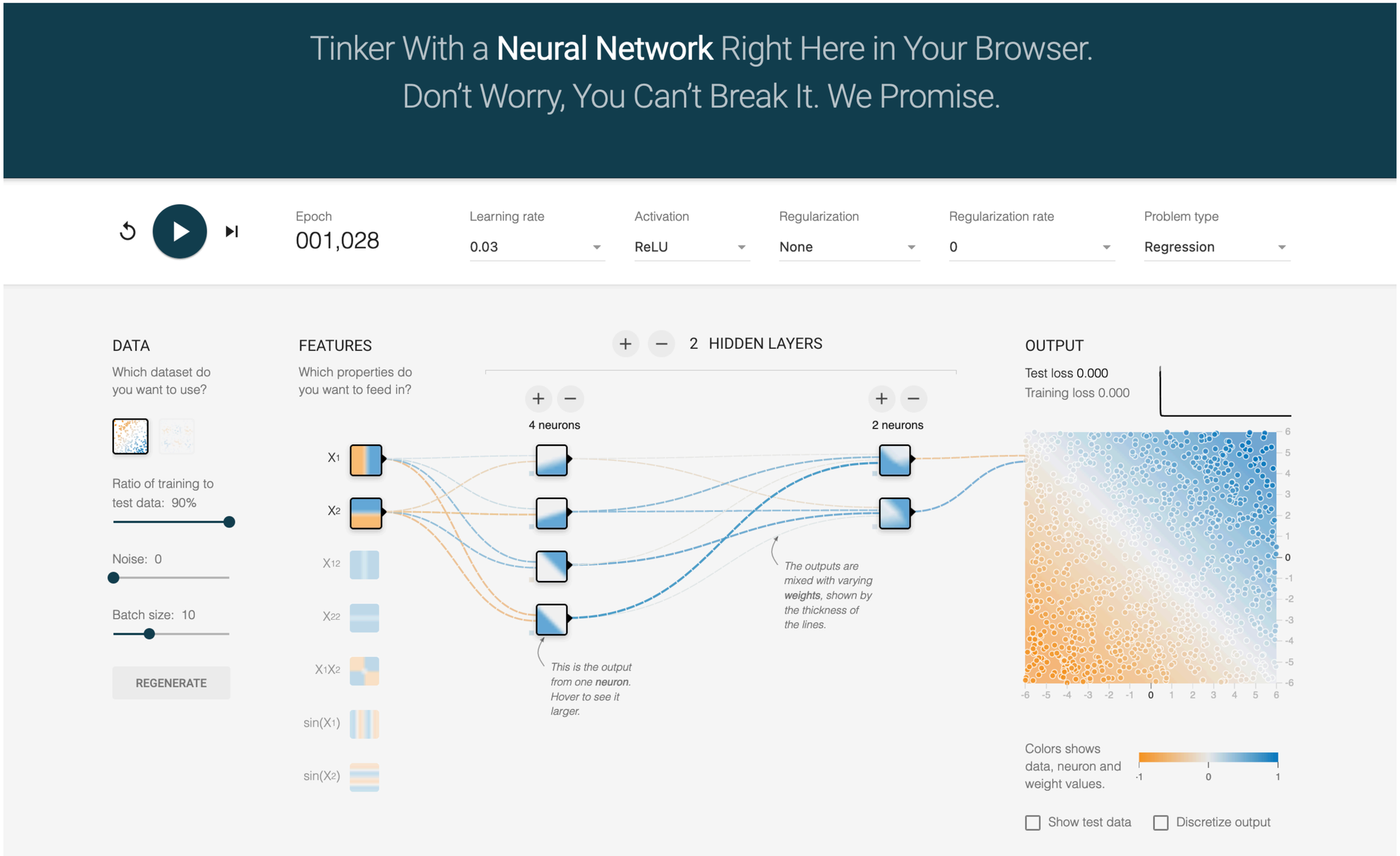
There’s no need to extract features from raw data (like sentiments in a sentence)











Playground[12]

Sigmoid

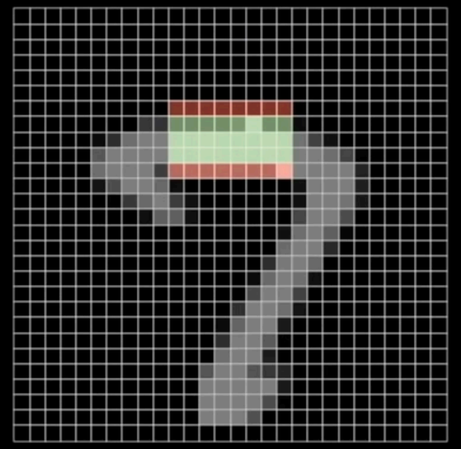
How positive is this?

$$\sigma(w_1 a_1 + w_2 a_2 + w_3 a_3 + \cdots + w_n a_n \boxed{-10})$$

"bias"

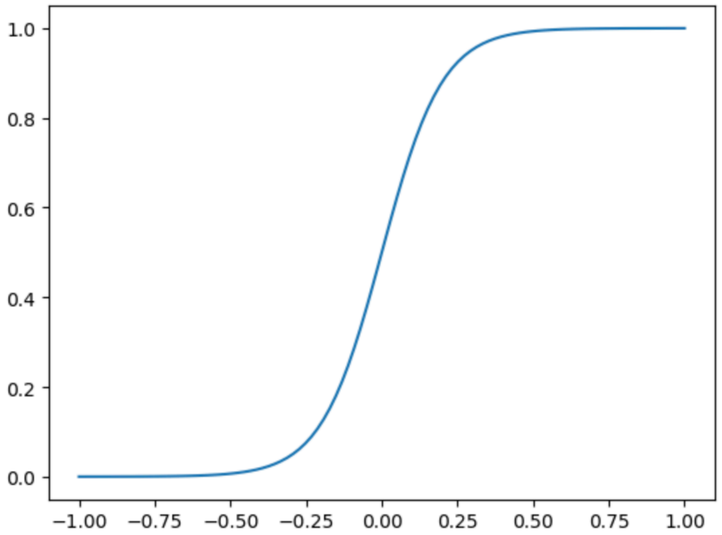
784

Only activate meaningfully when weighted sum > 10

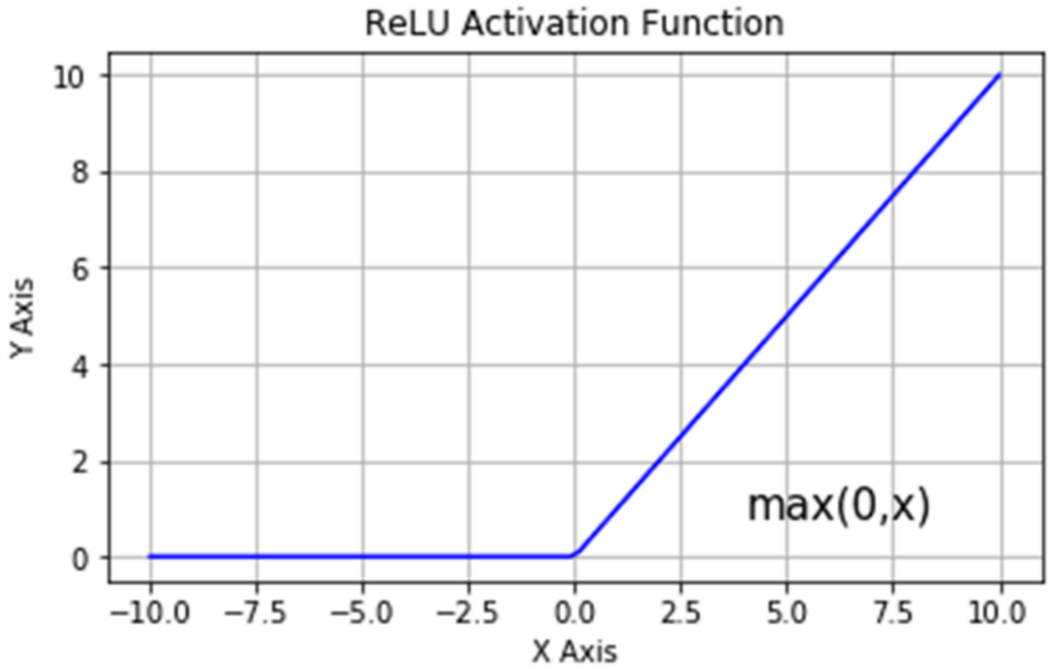


Activation Functions

Sigmoid Function



$$A = \frac{1}{1+e^{-x}}$$

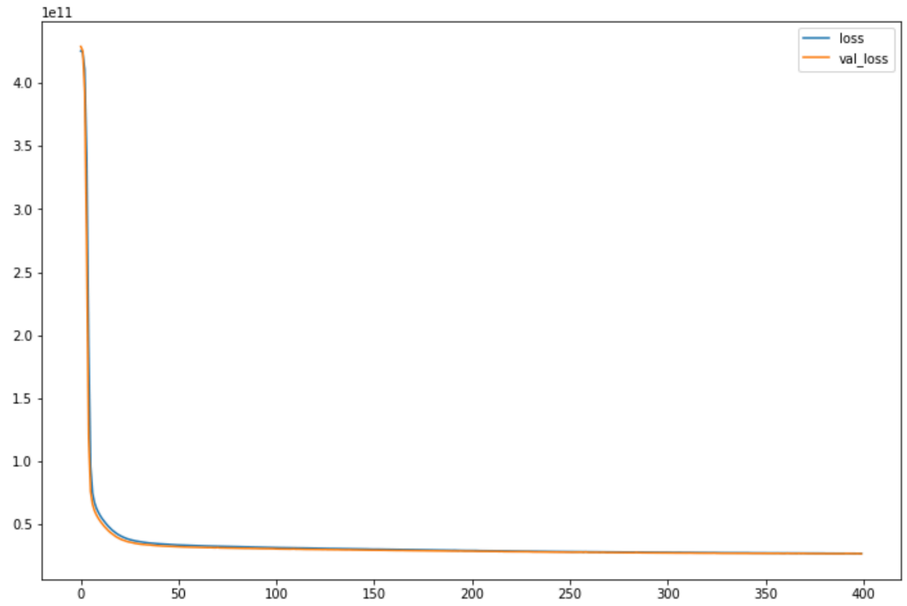


$$RELU(x) = \begin{cases} 0 \; if \; x < 0 \\ x \; if \; x >= 0 \end{cases}$$

The purpose of the activation function is to introduce non-linearity into the output of the neuron, allowing the neural network to learn complex patterns and relationships in the input data.

Implementation in Python

- Import Sequential, Dense, Activation & Adam (Adaptive Moment Estimation)
- Train the neural network
- Use fit( ) method to feed the training data into the network (x_train, y_train)
- Validate with the test datas (x_test, y_test)
- Declare batch size & epochs
- Plot the loss function with respect to training time
- Generate predictions
- Find accuracy & errors



The goal of training a neural network is to minimize the value of the loss function. This is done by adjusting the weights and biases of the network in a process called backpropagation.

```
156
157  """# Keras Regression"""
158
159  # Creating a Neural Network Model
160  from tensorflow.keras.models import Sequential
161  from tensorflow.keras.layers import Dense, Activation
162  from tensorflow.keras.optimizers import Adam
163
164  # having 19 nueron is based on the number of available featurs
165
166  model = Sequential()
167
168  model.add(Dense(19,activation='relu'))
169  model.add(Dense(19,activation='relu'))
170  model.add(Dense(19,activation='relu'))
171  model.add(Dense(19,activation='relu'))
172  model.add(Dense(1))
173
174  model.compile(optimizer='adam',loss='mse')
175
176  """## Training the mdoel"""
177
178  model.fit(x=X_train,y=y_train,
179            validation_data=(X_test,y_test),
180            batch_size=128,epochs=400)
181
182  model.summary()
183
184  loss_df = pd.DataFrame(model.history.history)
185  loss_df.plot(figsize=(12,8))
186
187  """### Evaluation on Test Data"""
188
189  y_pred = model.predict(X_test)
190
191  # evaluation metrics
192  # explained variance score: best possible score is 1 and lower values are worse
193  from sklearn import metrics
194
195  print('Mean Absolute Error: {:.2f}'.format(metrics.mean_absolute_error(y_test, y_pred)))
196  print('Mean Squared Error: {:.2f}'.format(metrics.mean_squared_error(y_test, y_pred)))
197  print('Root Mean Squared Error: {:.2f}'.format(np.sqrt(metrics.mean_squared_error(y_test, y_pred))))
198  print('Variance score is: {:.2f}'.format(metrics.explained_variance_score(y_test,y_pred)))
199
200  # we are off about 20% (comparing mean absolut error and mean of price)
201  Data['price'].mean()
```

Accuracy

Multiple Regression Model (70.4%)

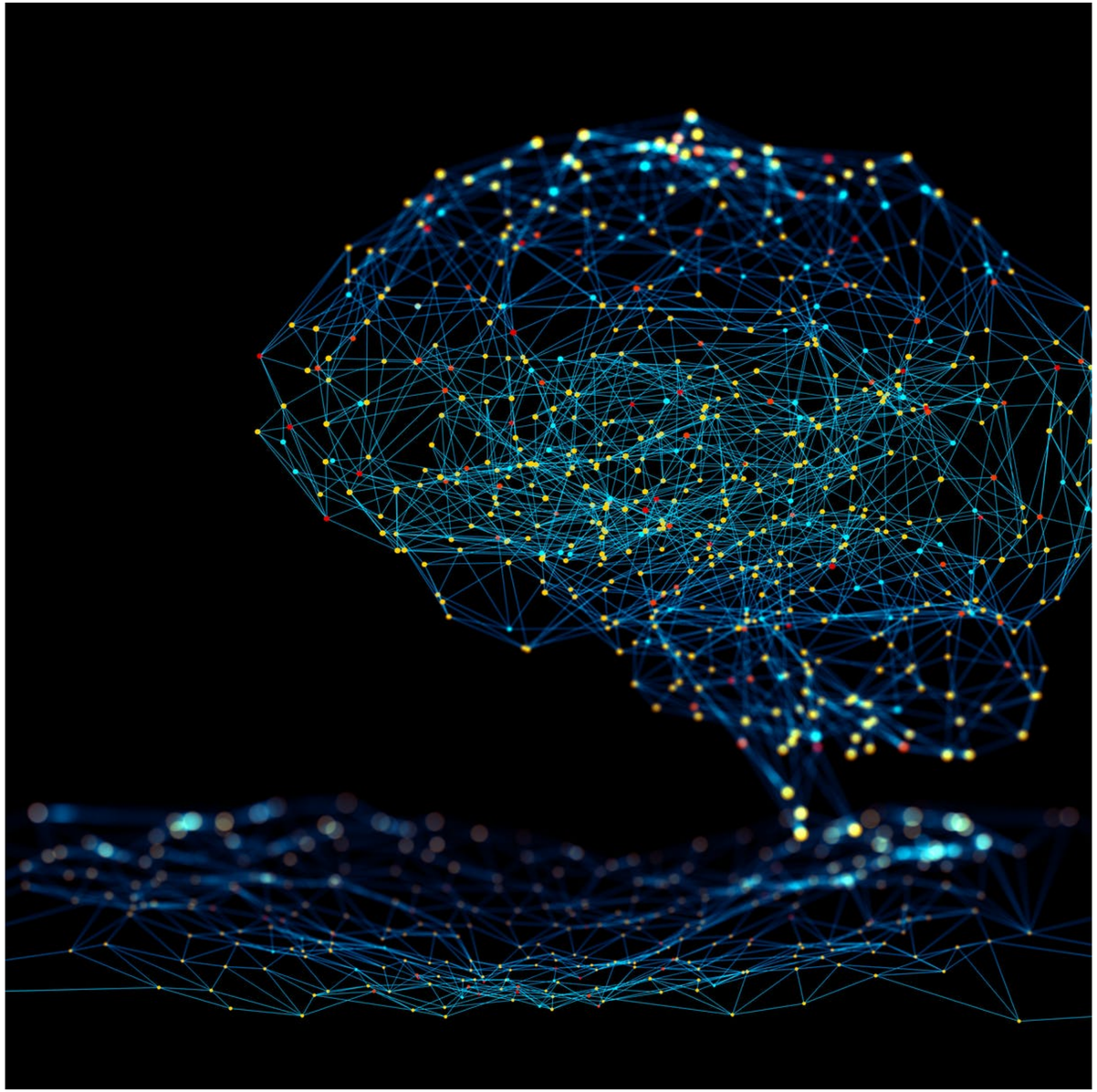
Neural Network using Keras (80.51%)

```
print('Model: Keras Regression\n')

print('Mean Absolute Error(MAE): {:.2f}'.format(metrics.mean_absolute_error(y_test, y_pred)))
print('Mean Squared Error(MSE): {:.2f}'.format(metrics.mean_squared_error(y_test, y_pred)))
print('Root Mean Squared Error(RMSE): {:.2f}'.format(np.sqrt(metrics.mean_squared_error(y_test, y_pred))))
print('Variance score: {:.2f}\n'.format(metrics.explained_variance_score(y_test,y_pred)*100))
print('**********************************\n')
print('Model: Multiple Linear Regression\n')
print('Mean Absolute Error(MAE): {:.2f}'.format(metrics.mean_absolute_error(y_test, y_predd)))
print('Mean Squared Error(MSE):{:.2f}'.format(metrics.mean_squared_error(y_test, y_predd)))
print('Root Mean Squared Error(RMSE):{:.2f}'.format(np.sqrt(metrics.mean_squared_error(y_test, y_predd))))
print('Variance score: {:.2f}'.format(metrics.explained_variance_score(y_test,y_predd)*100))
```

```
Model: Keras Regression

Mean Absolute Error(MAE): 101572.44
Mean Squared Error(MSE): 26737735523.03
Root Mean Squared Error(RMSE): 163516.77
Variance score: 80.51

**********************************

Model: Multiple Linear Regression

Mean Absolute Error(MAE): 125933.74
Mean Squared Error(MSE):40601153229.62
Root Mean Squared Error(RMSE):201497.28
Variance score: 70.40
```

Future Aspects

- Incorporating additional data sources, such as economic indicators or news articles, to improve the accuracy of the predictions.
- Experimenting with different neural network architectures, such as recurrent neural networks or transformers, to see if they perform better than the current architecture.
- Developing an online or real-time version of the improved model.
- Applying the model to different markets or assets, such as stocks, commodities, or currencies, to see if the model is applicable to other prediction tasks.



References

1. “*Deep Learning Basics: Introduction and Overview*”. MIT course 6.S094. Lex Fridman. https://youtu.be/O5xeyoRL95U
2. Quang Truong, Minh Nguyen, Hy Dang, Bo Mei, “*Housing Price Prediction via Improved Machine Learning Techniques”*, Procedia Computer Science, Volume 174, 2020, Pages 433-442, ISSN 1877-0509, https://doi.org/10.1016/j.procs.2020.06.111
3. Zulkifley, Nor & Rahman, Shuzlina & Nor Hasbiah, Ubaidullah & Ibrahim, Ismail. (2020). “*House Price Prediction using a Machine Learning Model: A Survey of Literature”*. International Journal of Modern Education and Computer Science. 12. 46-54, https://doi.org/10.5815/ijmecs.2020.06.04
4. Qingqi Zhang, "*Housing Price Prediction Based on Multiple Linear Regression*", *Scientific Programming*, vol. 2021, Article ID 7678931, 9 pages, 2021. https://doi.org/10.1155/2021/7678931
5. "*Neural Regression using Keras*”. https://visualstudiomagazine.com/articles/2018/07/23/neural-regression-using-keras.aspx
6. “*Regression with Keras”.* Pyimagesearch. https://pyimagesearch.com/2019/01/21/regression-with-keras/
7. “*House Sales in King County, USA”*. Kaggle. https://www.kaggle.com/datasets/harlfoxem/housesalesprediction
8. “*Multiple Linear Regression Model, A Quick Guide*”. Scribbr. https://www.scribbr.com/statistics/multiple-linear-regression/
9. “*But what is a neural network? | Chapter 1, Deep learning*”. 3Blue1Brown. https://youtu.be/aircAruvnKk
10. “*Matplotlib References*”. https://matplotlib.org/stable/api/index.html
11. “*Seaborn: Statistical Data Visualisation*”. https://seaborn.pydata.org/
12. “*A Neural Network Playground*”. Tensorflow. https://playground.tensorflow.org/

Thank You