HOUSE PRICES: ADVANCED REGRESSION TECHINQUES

I. Definiton:

Project overview: the challenge of this project is to produce a web-interface that will read user-defined inputs to predict the price of the user's house depending on the characteristics he/she has provided.
The project originates from a Kaggle competition (https://www.kaggle.com/c/house-prices-advanced-regression-techniques
) where the aim is to predict the house prices of the Ames's inhabitants (Iowa). The dataset used to train the predictive model has been provided by Kaggle itself.

Problem statement: the problem is to successfully predict the price of a given house according to its characteristics. The attempt to solve such an issue will be deployed in the following manner:

- Analyze the dataset to better understand the data and its distribution.
- Deploy a feature selection algorithm in order to break down the number of independent variables used to predict a house's price.
- Build and train a XXX Neural Network to make the predictions
- Create a web-interface providing 10 drop-down menus for the user to provide inputs describing the characteristics of his/her house.
- Link the trained NN in Sagemaker through using a Gateway API in order for it to receive the user inputs and make predictions upon them. Put a Lambda function in place to ingest the user input and transform it in such a way that predictions can be made and sent back.

Metrics: in order to analyze how successful, the predictions are, I am planning on using the Root Mean Squared Error (RMSE). I believe that this metric will adequately assess the goodness of the predictive model given that it gives a realistic and measurable error rate. RMSE comes in the unit of measure in which the dependent variable is measured (in this case $).

I believe this metric to be better for this purpose given that RMSE gives higher weights to large errors. Given that the aim of the project here is to have a relatively precise prediction, I think it would be a good idea to optimize for such a metric.
It is true that metrics like MAE are more responsive and robust with respect to outliers, however, it is also true that in the data analysis section I have not spotted any outlier that worries me, therefore, I cannot use such a positive characteristic of the MAE metric in my favor.

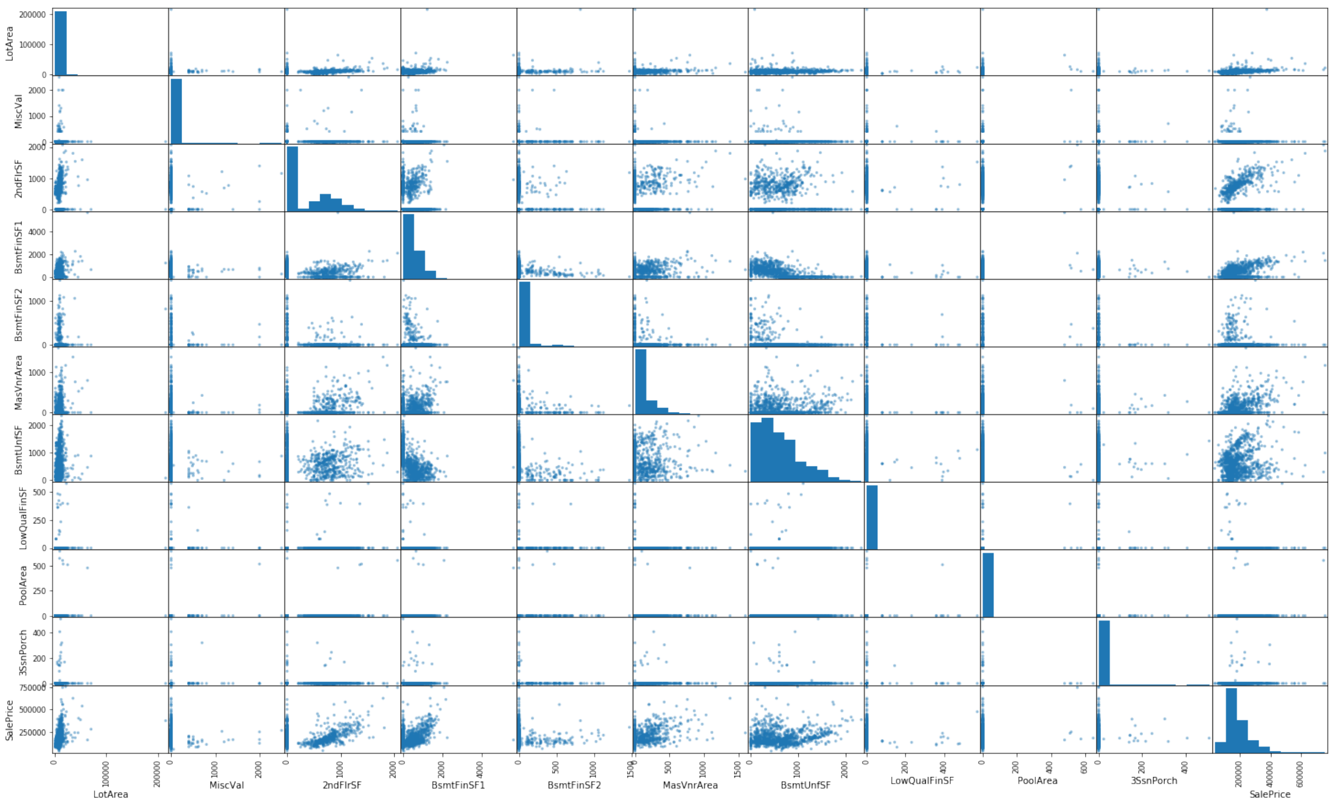
Benchmark:
Prior to deploying the aforementioned solution, I ran a linear regression on the same dataset that has been used for the purpose and the results in terms of MSE and RMSE are the following:

```
- MSE: 3038563741.3480158
- RMSE: 55123.1688253498
```

II. Analysis:

Data exploration:

- Scatter matrix

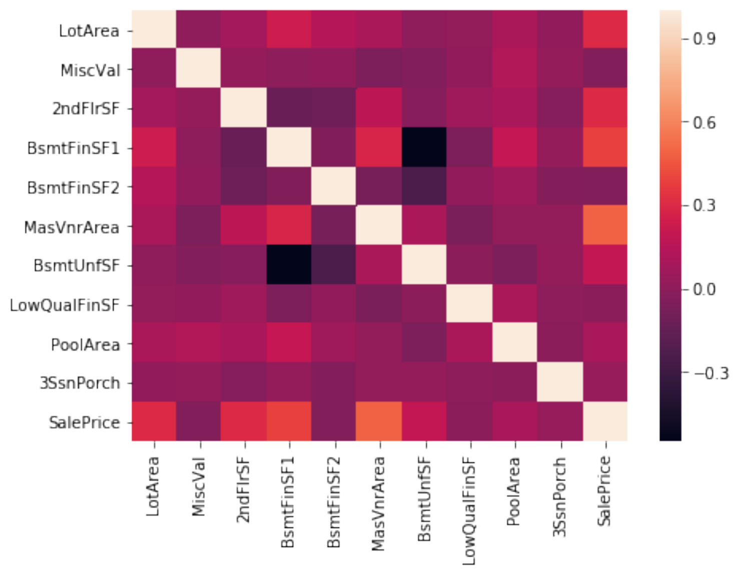


The scatter matrix above shows an exploration of the 10 variables with the strongest explicative power with respect to the dependent variable (SalePrice). The scatter matrix above shows each variable's distribution as well as a scatter plot exemplifying the correlation between each and every variable (amongst the selected ones).

It appears that the predominant distribution across the top 10 independent variables is the logarithmic one, where high frequencies of occurrences take place with low values of the respective variable. Said frequencies drop quite rapidly as measures of the respective variable increase.

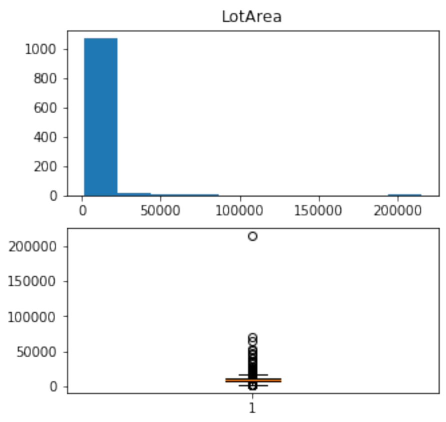
According to the scatter plots where the independent variables are plotted against the dependent one, the variables that show the strongest levels of positive correlation are: LotArea, 2ndFlrSF, BsmtFinSF1, MasVnrArea.

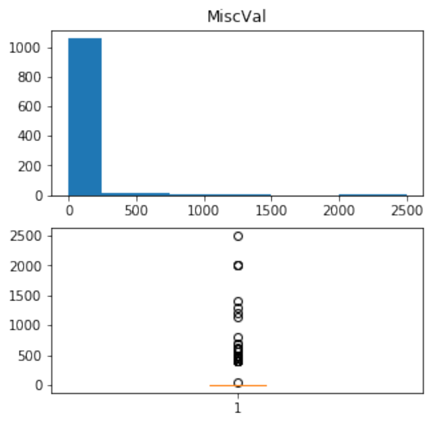
- Correlation Heatmap:

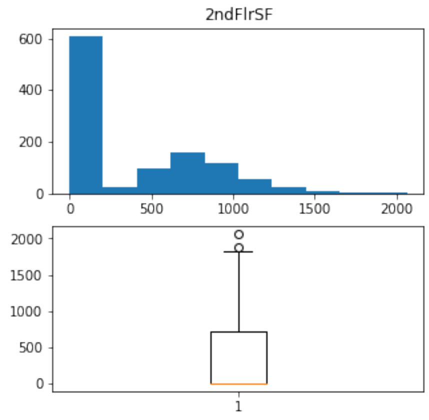


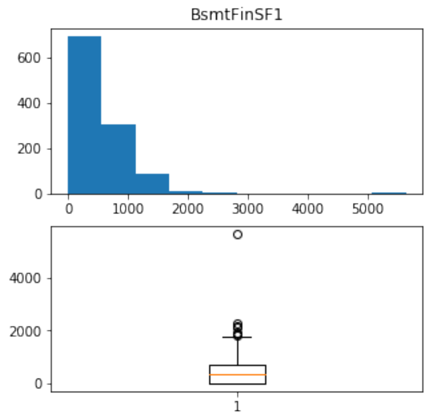
The correlation heatmap above reinforces the concepts depicted by the scatter matrix presented beforehand, with variables like: LotArea, 2ndFlrSF, BsmtFinSF1, MasVnrArea having the highest levels of correlation (above 0.3) with the dependent variable (SalePrice).

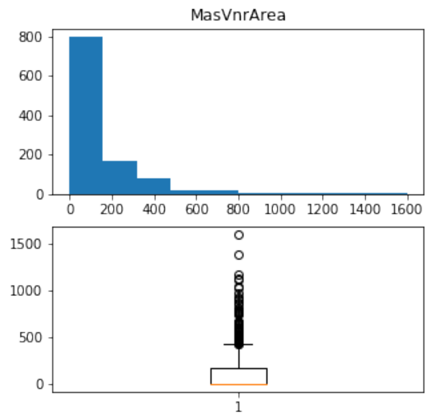
- Variable exploration:
    - The dataset has 1460 rows. This is a quite small dataset, however I think that this might be still a relevant and significant number of cases and th erefore I believe that the Neural Network will not be negatively affected by not having enough data points and finding itself stuck in a local mini ma. However, it is something to keep in mind.
    - With respect to missing data, there are very few variables that do not h ave 100% of the data represented. The dataset appears to be quite clea n, therefore I will not address the missing data part as an issue.

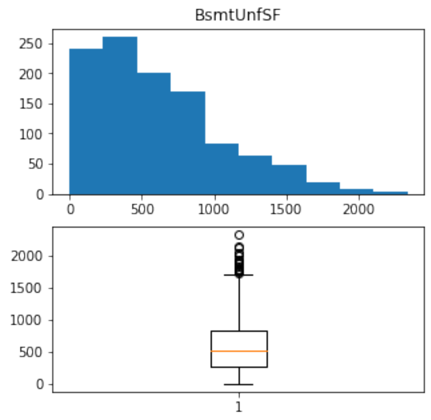


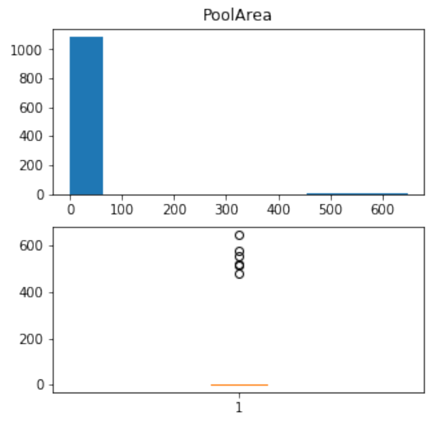


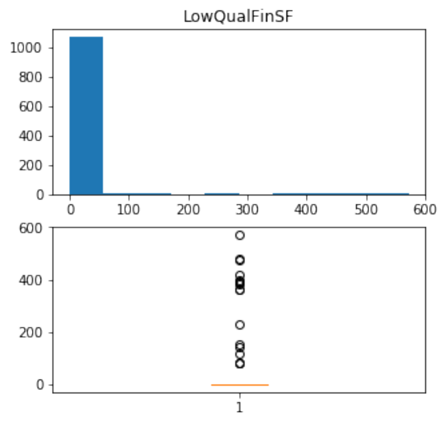

2ndFlrSF
0
200
400
600
0
500
1000
1500
2000
0
500
1000
1500
2000
1

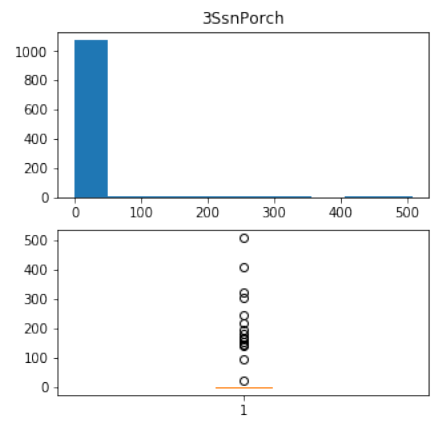

BsmtFinSF1
0
200
400
600
0
1000
2000
3000
4000
5000
0
2000
4000
1

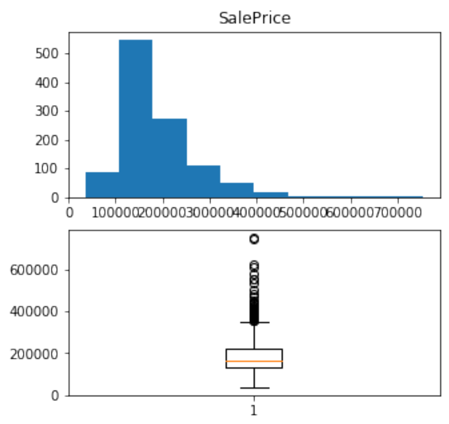

MasVnrArea
0
200
400
600
800
0
200
400
600
800
1000
1200
1400
1600
0
500
1000
1500
1


BsmtUnfSF
0
50
100
150
200
250
0
500
1000
1500
2000
0
500
1000
1500
2000
1


PoolArea
0
200
400
600
800
1000
0
100
200
300
400
500
600
0
200
400
600
1


LowQualFinSF
0
200
400
600
800
1000
0
100
200
300
400
500
600
0
200
400
600
1





Considerations:

- LotArea MiscVal, MasVrnArea, PoolArea, LowQualFinSF, 3SsnPorch appear to have a logarithmic distribution. Very high frequency of low values and dropping rapidly as the value measurement increases.
- BsmtUnfSF appears also to follow a logarithmic distribution, however, the decay appears to be slower than with the other variables that share the same trait.
- Treating variables which most of them follow a logarithmic distribution makes it more difficult for the algorithm to pick up the value of data points that are outside the big bulk of low value high frequency data point that characterize a logarithmic distribution.
- Sales price (the dependent variable) appears to follow a normal distribution, however being slightly skewed. The shape is not perfectly symmetric, as it appears to be a very fewer houses with a very high price and this concept makes total sense to me.

Algorithm & techniques:

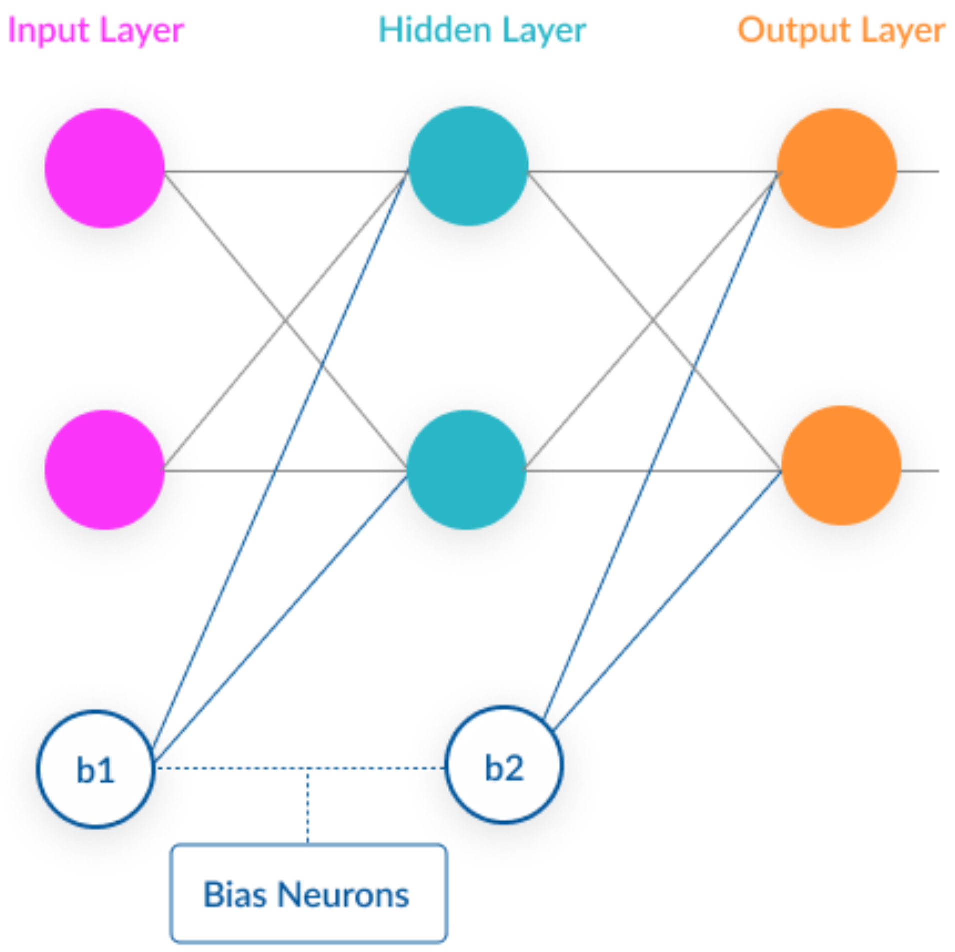
For this task I decided to put in action the knowledge I gained from the previous exercises of this Nano Degree and apply a Neural Network as a predictive algorithm for such a case.

Neural networks have proven to perform above average when dealing with predictive modeling tasks. The down-side of using a Neural Network is the fact that it works as a black box and therefore it is very difficult to understand what is actually happening inside it. Nowadays, few and very early-stage techniques for understanding what measures the algorithm puts in place to make a prediction are being developed, however, they are mostly apt for fields related to computer vision.
Given that I already took a deep dive with respect to the most important variables of the data set by firstly filtering the independent variables through an algorithm of Univariate Selection and secondly, inspecting each and every one of them through the visualizations above. This is why I decided to go forward with a Neural Network, given

that I feel I have the necessary context in order to go forward and interpret the results of the predictions made by such an algorithm.

Characteristics of a Neural Network:

1. Neural because its structure mimics the one of the brain.
2. Network because it's a number of strongly interconnected nodes (referred to as Neurons) separated across layers.
3. Data is introduced in input layer, which has 1 neuron for every component present in data.
4. Data is then manipulated across the hidden layers, this is where the job gets done. Weights and biases are the two components responsible for manipulating data.

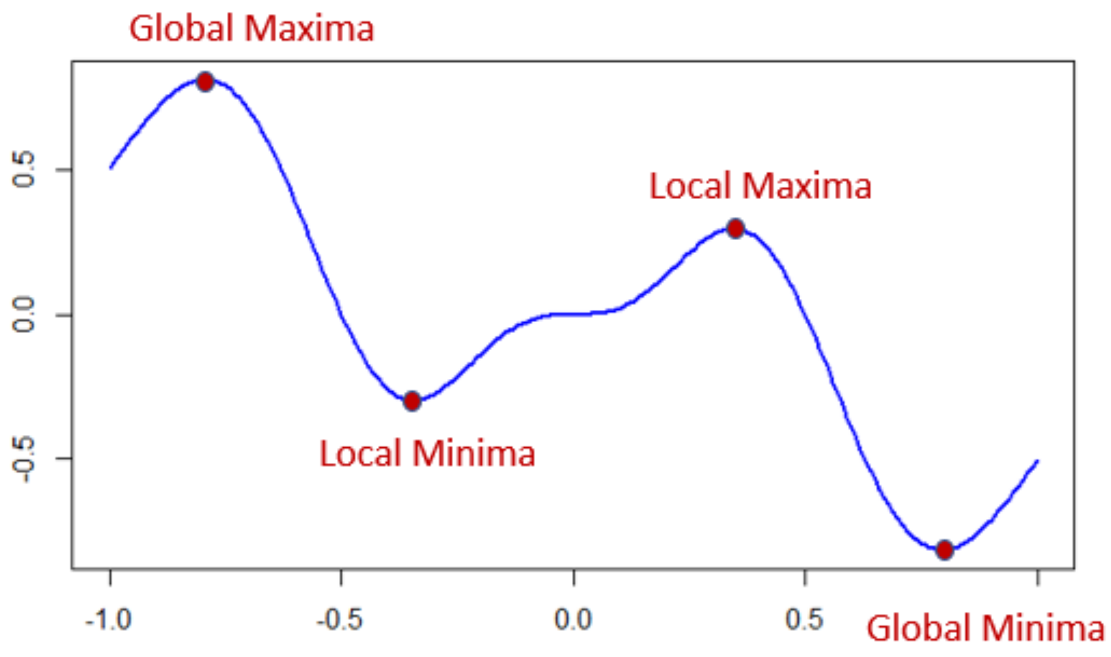


Bias unit is just appended to the start/end of the input and each hidden layer, and isn't influenced by the values in the previous layer

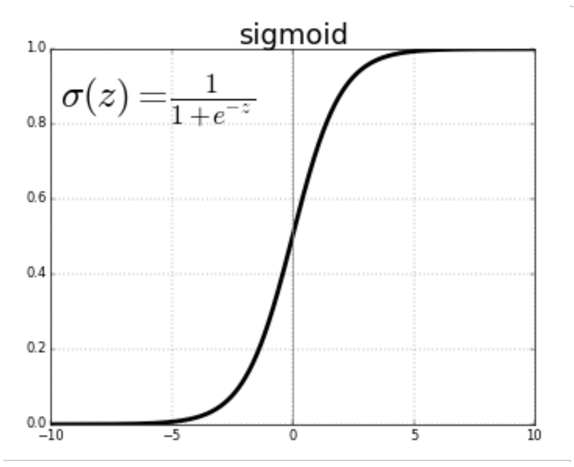
In the function y = mx + c , c is the bias. Without it the function would be forced to pass across the origin. Thus, bias helps in controlling the value at which activation function will trigger.

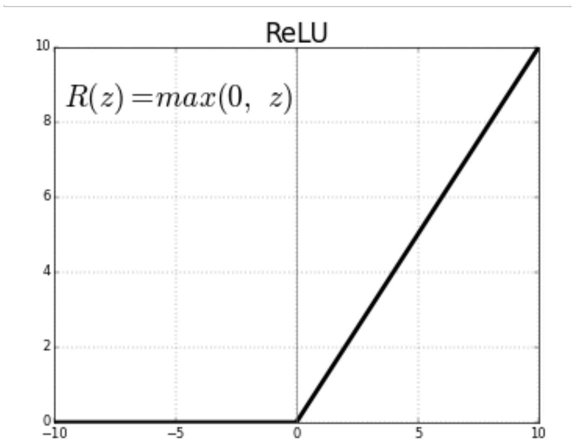
Weights are the mx in the y = mx + c. Its the strength of the connection between units

5. Input is ingested, weight & bias are calculated and result is passed down the connected layer of neurons (forward feed) until output layer.
6. Backward propagation: once output is obtained, the error is calculated by comparing actual vs. predicted value. The error is computed through a loss function.
7. Back-Propagation uses chain rule of Differential Calculus. In chain rule first we calculate the derivatives of error value with respect to the weight values of the last layer. We call these derivatives, gradients and use these gradient values to calculate the gradients of the second last layer. We repeat this process until we get gradients for each and every weight in our neural network. Then we subtract this gradient value from the weight value to reduce the error value.



8. Learning rate: Learning rate determines how quickly or how slowly you want to update your weight(parameter) values. Basically, how fast you want to go down the loss function.
9. Drop out - is a technique to avoid overfitting (patented by google). Drop out means that of every set of inputs (batch size), you eliminate a given % of those inputs. Why is dropout needed? —> A fully connected layer occupies most of the parameters, and hence, neurons develop co-dependency amongst each other during training which curbs the individual power of each neuron leading to over-fitting of training data.
10. Activation function:
  1. Relu (Rectified Linear Unit) : ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero.
  2. Sigmoid





III. Methodology:

Data processing: To find out which variables provide the highest level of explicative power in terms of variability with respect to the dependent variable, a feature selection algorithm has been applied. For this task I decided to use Univariate Selection.

The above plots depicting the distribution of every variable as well as a boxplot accounting for percentiles (25 & 75) as well as mean and outliers, have lead me to take into consideration the following hypothesis: the logarithmic distribution justifies a strong number of data points at low levels of measurement for each variable, and therefore the less frequent, high values for each and every one of those variables may seem outliers at first, however I believe that by removing them important data points storing valuable information would be removed from the dataset thus reducing the variability in the data which may produce better results in the training set, however, affecting negatively the performance of the model's generalization power (against the test set). Keeping this in mind, this is why I decided to keep these points who look like outliers when analyzing the box plots.

In order to go forward with the exercise, I split the entire dataset according to an 80-20 split, where 80% of the data has been used for training purposes and 20% of the data has been used for testing purposes.

Implementation:

The implemented solution has managed the data from its source in the following way:

- Feature Selection: use Univariate Selection algorithm to pick the top 10 variables with the strongest predictive power.
- Anomaly inspection: these top 10 variables have been thoroughly inspected for any outliers or abnormalities
- Training/Predictions: an XXX Neural Network has been built, and trained according to which hyperparameters delivered the best results.

- Web-interface generation: create a simple web page and through the use of drop-down menus store user-defined inputs that will display the characteristics of a user's house.
- Lambda function to ingest user input: put in place a Lambda function that will process the user inputs and call the NN endpoint in order to make predictions. The lambda function will also deliver said predictions.
- API Gateway: API used to connect the web-interface with the model and lambda in order to deliver the predictions made.

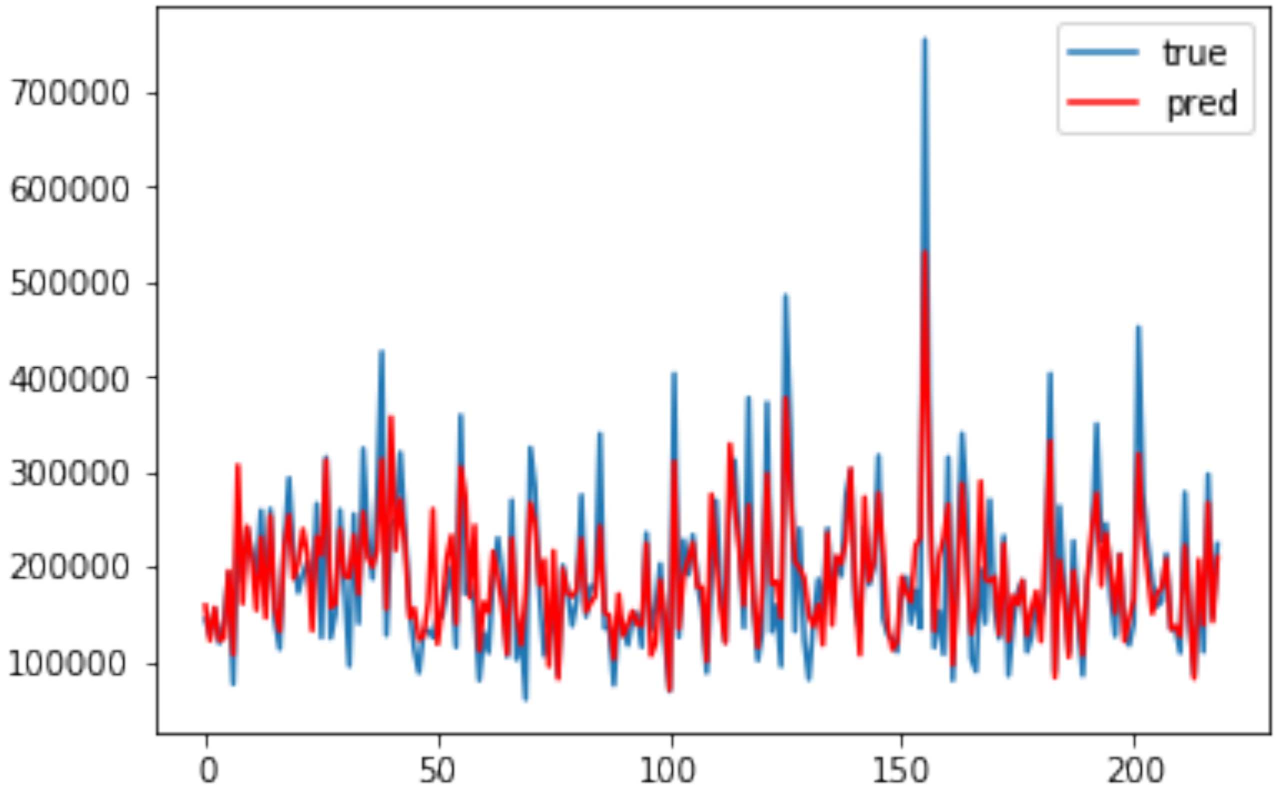
Refinement: the process of refinement put into practice has been the manual adjustments made to the Neural Network's architecture by tweaking both its structure (layers, neurons ...etc.) as well as its hyperparameters in order to optimize for loss during the training phase.
After various training phases, an optimal solution has been reached. The Neural Network with the following characteristics has been put in place:

- 2 layers
- 128 neurons
- Loss function – MSE
- Number of epochs – 200
- Loss rate – 0.01

I have not put in place the hyperparameter optimization technique provided by Sagemaker given that I built this NN from scratch and wanted to understand how each hyperparameter as well as component of its architecture would affect the training phase. This is why I did not implement the automated command the picks the best hyperparameters for you, but I went for a "manual" and more mechanical version of it (trial and error).

IV. **Results:**

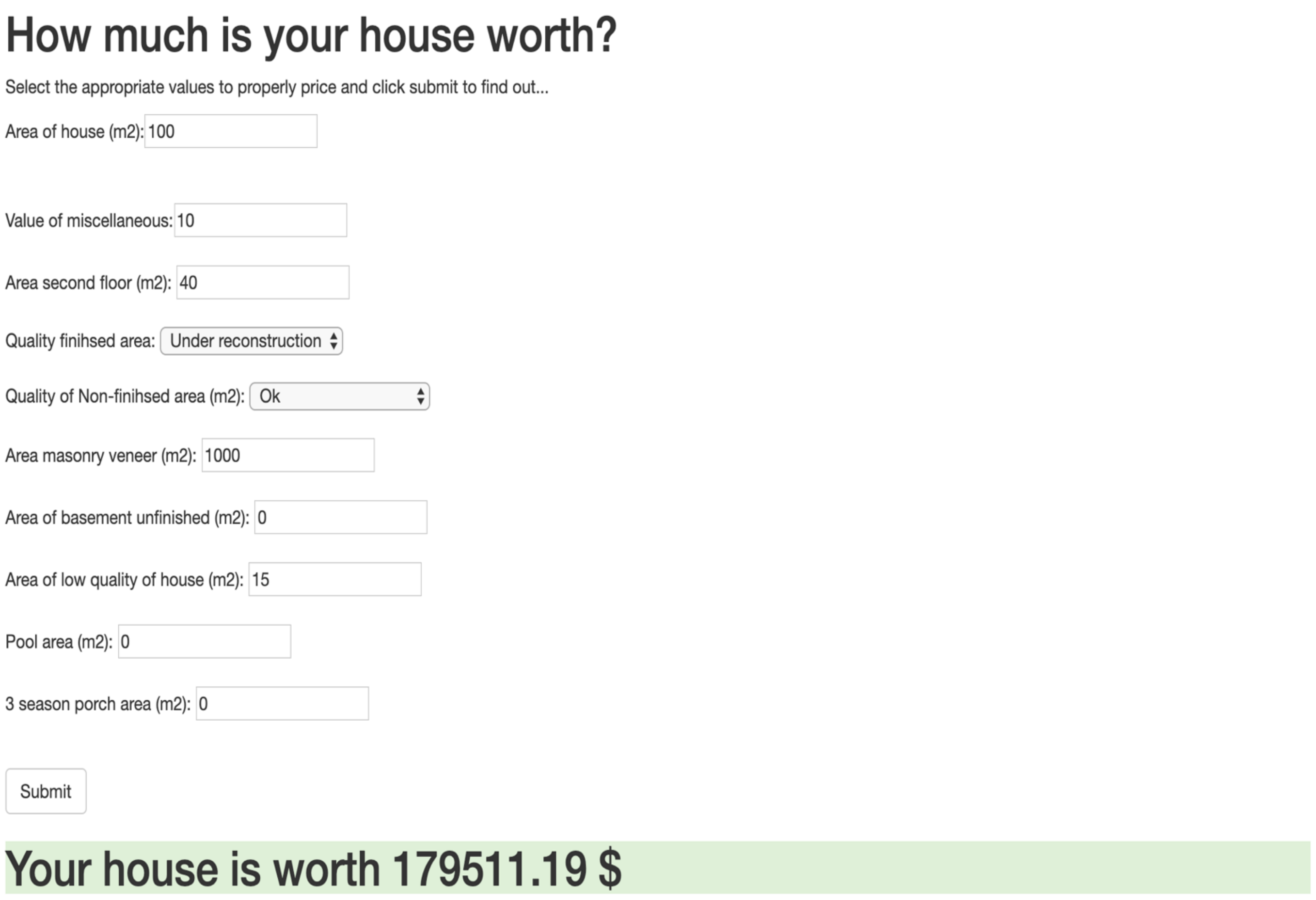
Model evaluation and validation:



```
- MSE: 2,086,653,985.72
- RMSE: 45,679.91
```

Looking at the graph above it is clear how the deployed model is performing quite well with houses that tend to float around a price of 200K, however, it looks like it doesn't pick up houses with especially high prices (>= 400K).
This model appears to be a better implementation than a simple linear regression give n that on average the RMSE is 10K lower.
An improvement would be to look in the determining factors that make the price of th ese houses so high and find ways to make it easier for the model to pick up such a signi ficant trait.
Clustering is an option.
PCA is also an option considering.

V. **Conclusion:**



How much is your house worth?

Select the appropriate values to properly price and click submit to find out...

Area of house (m2):	100
Value of miscellaneous:	10
Area second floor (m2):	40
Quality finihsed area:	Under reconstruction
Quality of Non-finihsed area (m2):	Ok
Area masonry veneer (m2):	1000
Area of basement unfinished (m2):	0
Area of low quality of house (m2):	15
Pool area (m2):	0
3 season porch area (m2):	0

Your house is worth 179511.19 $

This visualization is to show how a first version of the end to end solution works. The idea of this project was to give an end user a quick and easy way to find out its house's worth.

Reflection:

The most interesting aspect of the project in m phase opinion was to make the Lambda function work. It was amazing to see how the inputs went from the frontend to the Lambda that was linked with the Neural Network endpoint that made the prediction and sent it back to the front end.

Improvement:

This final submission fits my expectations because the skeleton sitting behind this infrastructure is solid and works really well.
The next steps I would apply to this project are the following:

1. In-depth analysis of the model's failures to find ways to improve it (clustering, PCA are just initial ideas)
2. Improve the front-end, in order to give the user a better UI/UX experience due to the look and feel of the app.

It would be interesting to see how people would interact with such an app and maybe integrate it with some other functionality like:

- Exemplifying the characteristics of houses with similar house prices
- Adding a location functionality – geo location (latitude and longitude) features are always interesting and very visually appealing in an app.