***Project Report***

*Group B*

*Pytorch & Pyplot*

*Submitted By*

*Sonnet Padayatty Reju*

*Student ID: 18200682*

*MSc. Data & Computational Science*

**Chapter 1**

**Introduction**

* 1. **Project Description**

We are analyzing a dataset with following packages

* Pytorch : For Deep Learning
* Plotly : For Visualization

Our aim is to predict the house price from King County, USA. As the desired output is a continuous variable, we need to adapt regression machine learning methods for the analysis part.

**1.2 Problem Statement**

The dataset consisted of historic data of houses sold between May 2014 to May 2015. The dataset consists of house prices from King County an area in the US State of Washington.

**Data fields**

**Input variables:-**

* id
* date
* bedrooms
* bathrooms
* sqft\_living
* sqft\_lot
* floors
* waterfront
* view
* condition
* grade
* sqft\_above
* sqft\_basement
* yr\_built
* yr\_renovated
* zipcode
* lat
* long
* sqft\_living15
* sqft\_lot15

**Target Variable**

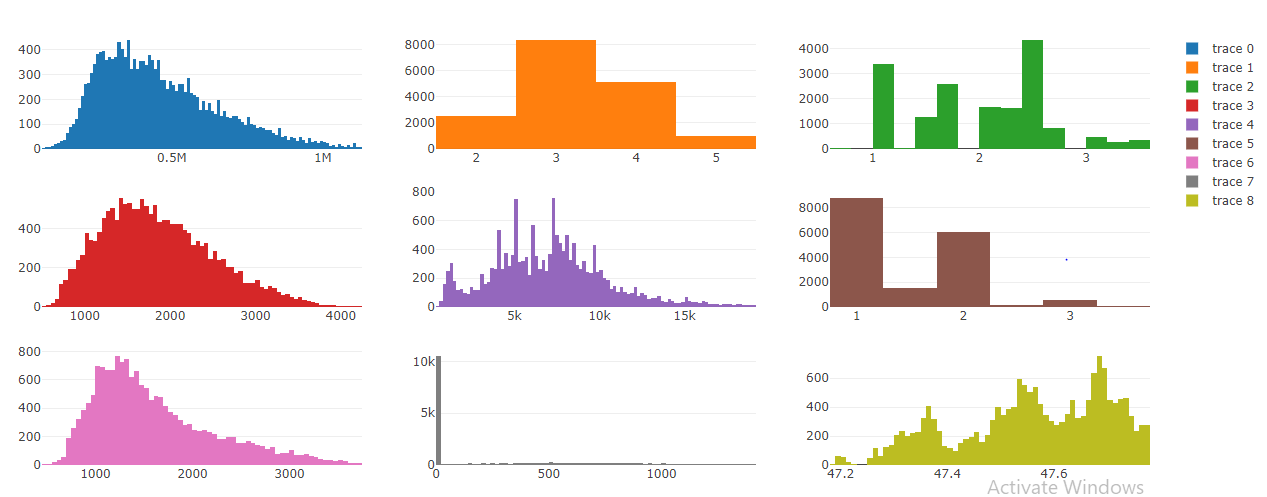
* price

**Chapter 2**

**Methodology**

**2.1 Pre-Processing**

Pre-processing of data is an indispensable stage in predictive analysis. Since the predictive model needs to handle a big data set, it is always necessary to eliminate unwanted data. There may be many variables whose data type is incorrect and may create complexity while training the predictive model with train dataset. In order to minimize such issues in modeling stage, we conduct data pre-processing and extract important insights from the raw data. We can extract such information by analyzing the independent variables using probability density function or by visualizing how the data points have been distributed in each variable. It can be easily achieved by normality checking functions like histograms as shown below

.

Following are main pre-processing methods used in predictive analysis

**2.1.1 Missing Value Analysis**

Once we are done with pre-processing steps like “Renaming the variables” and “Converting into proper Data types”, we can conduct the missing value analysis. You may use the combination of both train and test data for the imputation of missing values as it makes the model to predict the values more accurately. Mainly, there are 3 methods for imputation of missing values.

1. Mean method

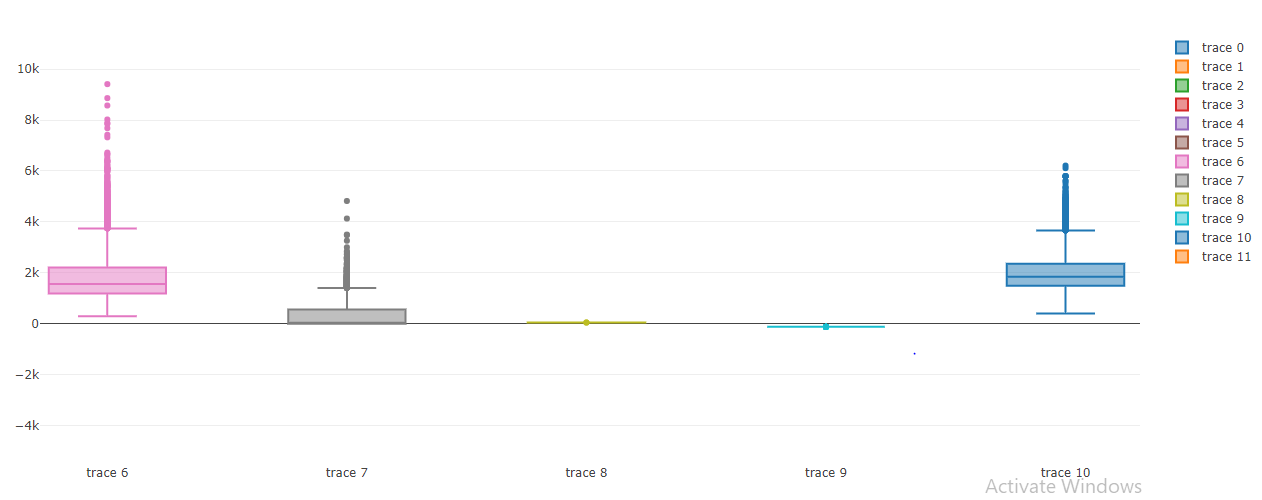
2. Median method

3. KNN imputation

We are not supposed to do imputation if the percentage of missing values in a variable is more than 30%. Since none of the variables exceed 30%, we can proceed with imputation. From the analysis, it is found that KNN imputation is the effective technique for this dataset.

**2.1.2 Outlier Analysis**

By definition, outliers are points that are distant from remaining observations. As a result, they can potentially skew or bias any analysis performed on the dataset. It is therefore very important to detect and adequately deal with outliers. In order to show the impact of outliers, we use a technique called box plot in which the distribution of data points is visualized as shown below.



The outliers that we found in boxplot cannot always be considered as irrelevant data. The outliers could be relevant if we found that the prediction power is dependent on them. In such cases, outliers cannot be considered as data entry errors, so it is important that we need to keep those data in model creation. In our dataset, we found that the removal of outliers doesn’t make much difference in Accuracy of the model.

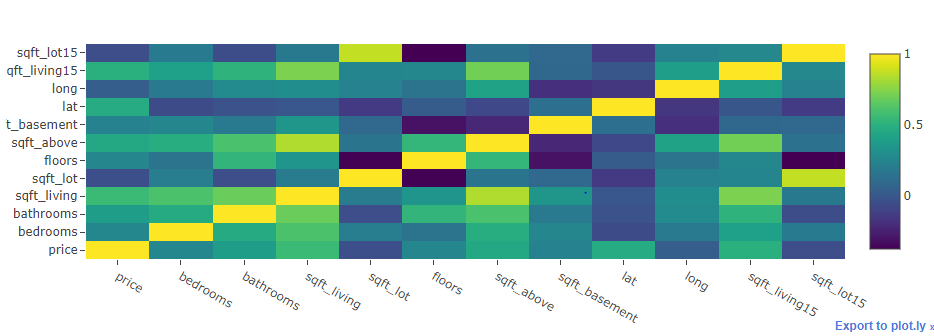
If we are in position that the removal of outliers is mandatory, then we can replace those values by NaN and do the step - KNN imputation again. The distribution of data became more normal / less skewed after the removal of outliers. This illustrates that the removal of outliers from the data may improve the accuracy of the predictive model in some cases.

**2.1.3 Feature Selection**

Feature selection is extremely important in machine learning primarily because it serves as a fundamental technique to direct the use of variables to what's most efficient and effective for a given machine learning system. It helps to minimize the curse of dimensionality or help deal with over fitting feature selection helps to give developers the tools to use only the most relevant and useful data in machine learning training sets, which dramatically reduces costs and data volume. There are many ways to do feature selection, but in this project we use Chi-Square test and Correlation Analysis for the feature selection of Categorical and Continuous variables respectively.

The correlation plot of numerical variables is shown below. From the figure, it is clear that the following variables are highly correlated to each other.

1. Sqft\_lot\_15 & Sqft\_lot
2. Sqft\_living\_15 & Sqft\_living
3. Sqft\_above & Sqft\_living



**Correlation plot:**-

When it comes to categorical variables, we need to perform Chi-Square test of Independence in order to extract unwanted variables from the data set. The result of Chi-Square test is shown below.

id

0.9999980648526372

date

4.968185512829421e-16

waterfront

3.3617488288706776e-37

view

1.4383342423329998e-16

condition

4.353581408408281e-83

grade

0.0

yr\_built

1.0

yr\_renovated

1.1600724930992988e-101

zipcode

2.196968315959122e-106

Since the p values of the variables- “id” "grade" and "yr\_built" are greater than 0.05 or not significant, it is found that these variables can be eliminated from modeling stage.

Therefore, as part of Dimension reduction, the following variables are removed from the data set.

Variables: ‘yr\_built', 'grade', 'id', 'sqft\_living15', 'sqft\_lot15', 'sqft\_above'

**2.1.4 Feature Scaling**

Feature scaling is a method used to standardize the range of independent variables or features of data. In data processing, it is also known as data normalization. Since the range of values of raw data varies widely, in some machine learning algorithms, objective functions will not work properly without normalization. For example, the majority of classifiers calculate the distance between two points by the Euclidean distance. If one of the features has a broad range of values, the distance will be governed by this particular feature. Therefore, the range of all features should be normalized so that each feature contributes approximately proportionately to the final distance. Another reason why feature scaling is applied is that gradient descent converges much faster with feature scaling than without it.

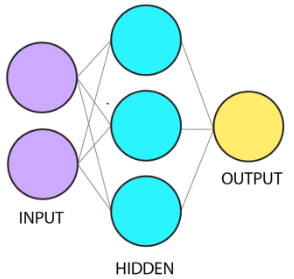
In this project, we apply feature scaling method- Normalization.

**Chapter 3**

**Modeling**

For the model creation, we use the package – **Pytorch**.

Pytorch is a python library for developing and evaluating deep learning models. It allows us to define and train neural networks in a few short lines of code. Similar to Keras, we are building a computation graph that dictate the flow of data and what type of operations need to be performed. Please see the graph below



The model we selected in our case is Sequential and the model creation comprises of following steps.

1. **Loading the data**

We are already done with this step as mentioned above.

1. **Splitting the dataset**

The dataset is spitted into 14 input variables (X) and 1 output variable (Y).

1. **Model creation**

Creation of model can be done by sequence of layers after importing nn.sequential() and nn.Linear(). We create sequential model by adding layers in our network topology. The efficient topology can be found by trial and error method. Generally, we need a network large enough to capture the structure of the problem.

Here, we use a network structure with 3 layers as follows

model = nn.Sequential(nn.Linear(n\_in, n\_h3),

nn.ReLU(),

nn.Linear(n\_h3,n\_out),

nn.Sigmoid())

The layers are defined using Linear functions. We can specify input dimension(n\_in) as first argument, and hidden layer as the second one (n\_h3).

In this case, weight initialization step is carried by default.

We will use relu activation function and sigmoid function in the sequential function in order to perform the forward propagation.

1. **Construct the loss function**

To find out the loss of prediction in each epoch, we need to define a loss function. As it is regression problem, we need to use MSELoss function as shown below.

criterion = torch.nn.MSELoss()

1. **Construct a optimizer**

Torch.optim is a module that implements various optimization algorithms used for building neural networks. Here, the optimizer used is Adam and the learning rate is defined as 0.001.

optimizer = torch.optim.Adam(model.parameters(), lr=0.001)

1. **Gradient Descent**

In simple language we can say that, it is the process which helps to find the best weights that can minimize the loss over several iterations or epochs. The steps that we follow to find best weights are follows:-

**Forward pass:** Compute predicted output (Y) by passing the input (X) to the model.

y\_pred = model(X)

**Loss :** Compute loss by calling the loss function criterion.

loss = criterion(y\_pred, y)

**Zero gradients :** Performs a backpass and update the weights using the below code

optimizer.zero\_grad()

**Back propagation:**

loss.backward()

**Update the parameters**

optimizer.step()

1. **Predict the output of new data**

This can be achieved by putting the test input data into our trained model as shown below.

prediction=model(X1)

**If we want the actual predicted values, instead of normalized values. We need to use inverse transform function as shown below.**

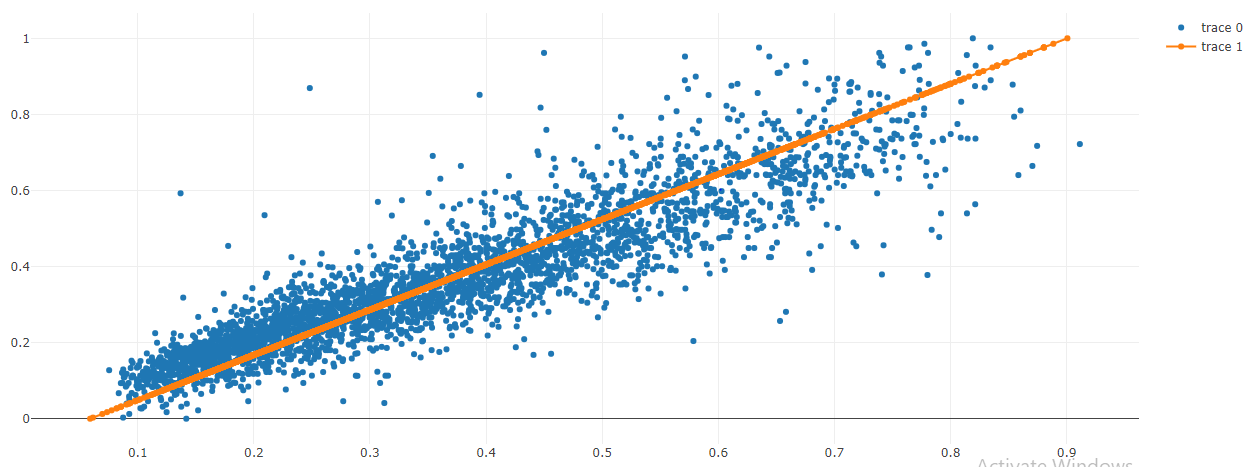
y\_pred\_act = min\_max\_scaler.inverse\_transform(pred\_arr)

**Chapter 4**

**Conclusion**

The minimum loss of the trained model is **0.000263**

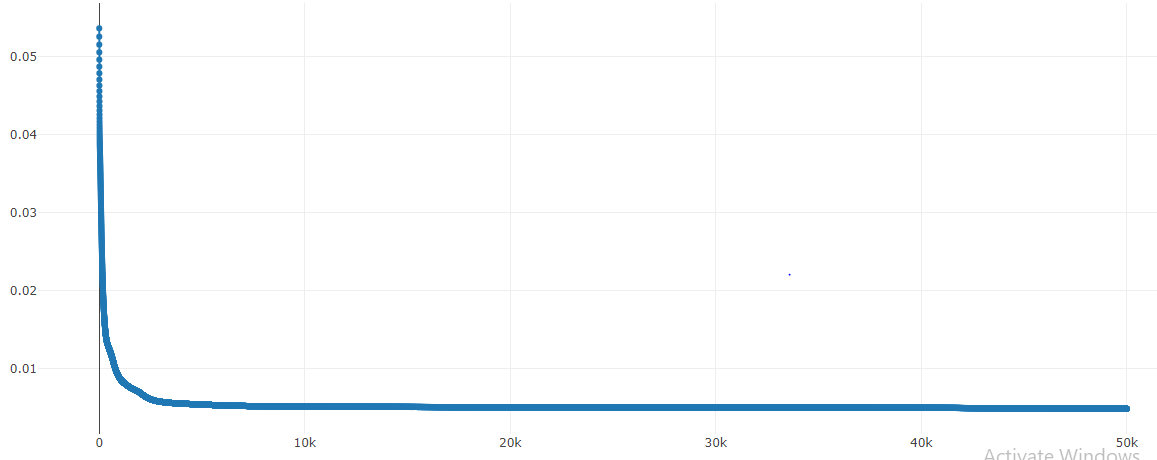
The loss we found after passing a new data/ test data is **0.0009**



**.**

**The graph of Actual values vs Predicted values**

**.**



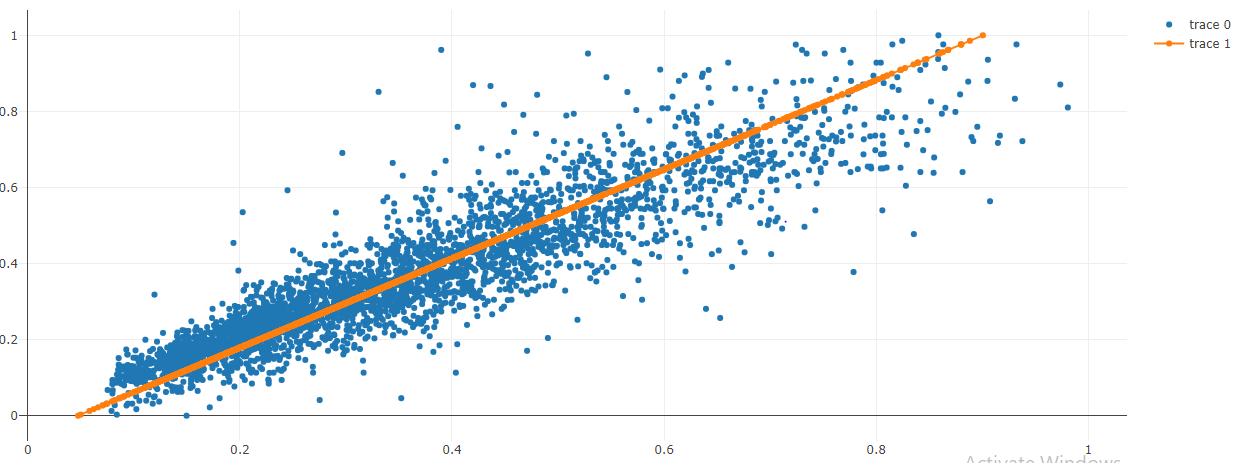
**The graph of Epoch vs Loss**

**Via Keras**

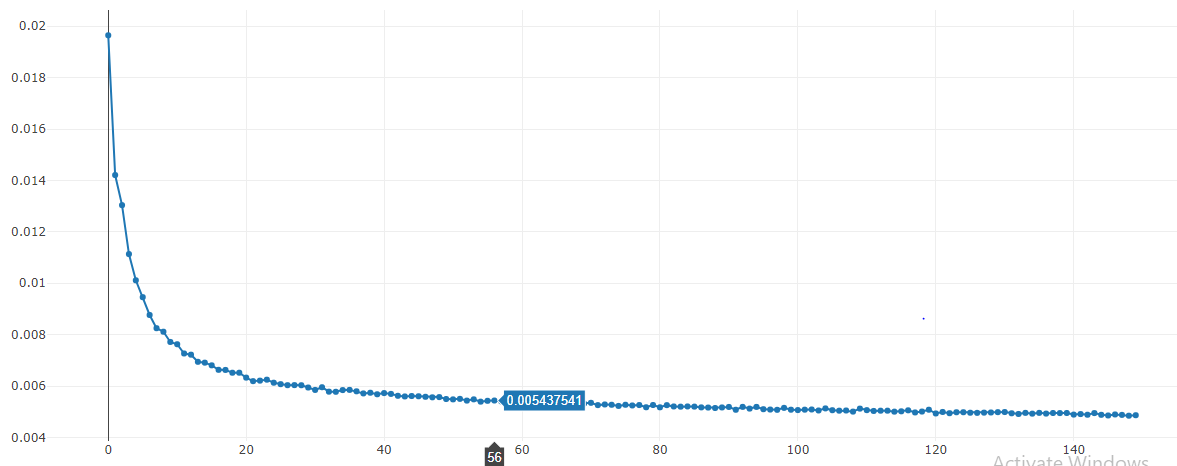
The loss obtained for trained model is **0.00035**

The loss obtained for test dataset is **0.00053**

**Hence , in this dataset, even though the model built via Pytorch performs better in training dataset, we found that the model created through Keras has lower loss while predicting the outputs of test dataset.**



From the above graphs, it is clear that the fitted line in the Keras model accommodates more data points in minimum epochs compared to the Pytorch model.



**The graph of Epoch vs Loss**