REPORT

Lab Assignment 10

Vrushali Pandit - B20BB047

TASK 1:

Data:

The dataset contains different features of a flower such as - sepal length, sepal width, petal length and petal width. The target variable is the species of flower {iris-setosa, iris-versicolor, iris-virginica} . Using the different features, we have to predict the target species.

Pre processing:

- The target variable is a string categorical variable and must be converted to a numeric one. We use **Label Encoding** for the same.
- Now, **split** the dataset into 70:30 ratio of train:test
- Because Multi Layer Perceptron uses gradient descent we must **standardize** the features. Standard scaling removes any bias among the features.

Part a): Vary the learning rate for a fixed size of hidden layers and show the best learning rate value when you run it for 50 epochs

- Used a for loop to vary the learning rate alpha = {0.001,0.02,0.03,0.04,0.01,0.1,1} for a MLP with two hidden layers and relu activation. the 1st hidden layer has 150 nodes and the 2nd has 100 nodes.
- By running the same code multiple times we get different results because sklearn's MLPClassifier uses stochastic gradient descent. We must set random_state = 0 to get reproducible results.

alpha	Test accuracy
0.0001	0.933
0.001	0.933
0.01	0.933
0.1	0.933

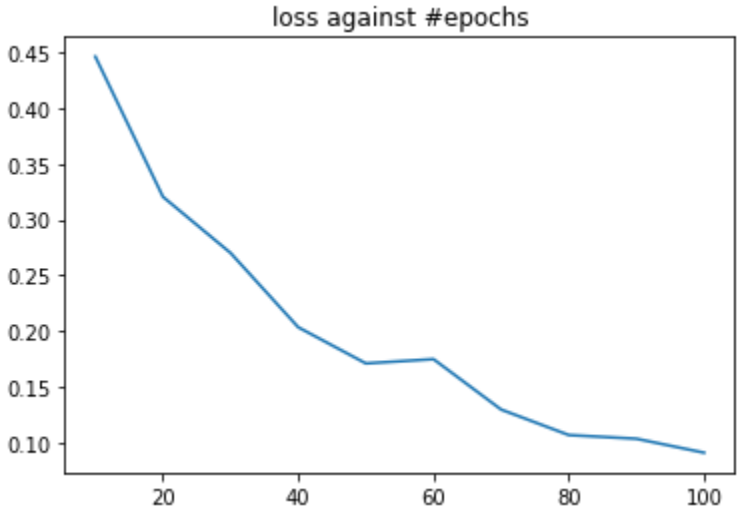
1	0.888
10	0.822

Why?

- The question mentions to use #epochs = 50. This means (confirmed by warning) that the process gets over before convergence. This may lead to constant accuracy over a range of values.
- A lower accuracy for alpha = 1,10 -> really large values for learning rate. The process gets over way before convergence. This is because the step sizes are larger, and it keeps oscillating around the optimum a lot.

Part b): Vary the number of epochs from 10-100 in a step of 10 and show the loss value curve, using the best set of hyperparameters.

- Using GridSearchCV is found the best set of hyper parameters where activation varied on ['relu','sigmoid','tanh','identity'], and the learning rate alpha varied on [0.001,0.01,0.1]
- The best parameters return an MLPclassifier with:
  - **2 hidden layers**
  - **Number of nodes in 1st, 2nd layer = 150,100**
  - **Activation = tanh**
  - **Alpha = 0.001**
- Now we loop #epochs from 10 to 100 in steps of 10 and plot the loss value at each iteration.



Why?

- For an increasing number of epochs, we have decreasing loss.
- Epoch means iteration over the entire dataset. Because MLP uses gradient descent for optimization.
- The iterative quality of the gradient descent helps a under-fitted graph to make the graph fit optimally to the data.
- Because it is an iterative algorithm, the more number of iterations/epochs the lower the loss.

TASK 2:

Data:

- The dataset contains information about houses' features and their corresponding values. Size = 1459 samples and 79 features and 1 target variable.
- Some of the features are Neighborhood, Lot area, Year of selling, basement quality etc.
- The features include all types of features such as temporal (time based), numerical discrete, numerical continuous and categorical.

Pre-processing and Data Cleaning:

Done in the following steps:

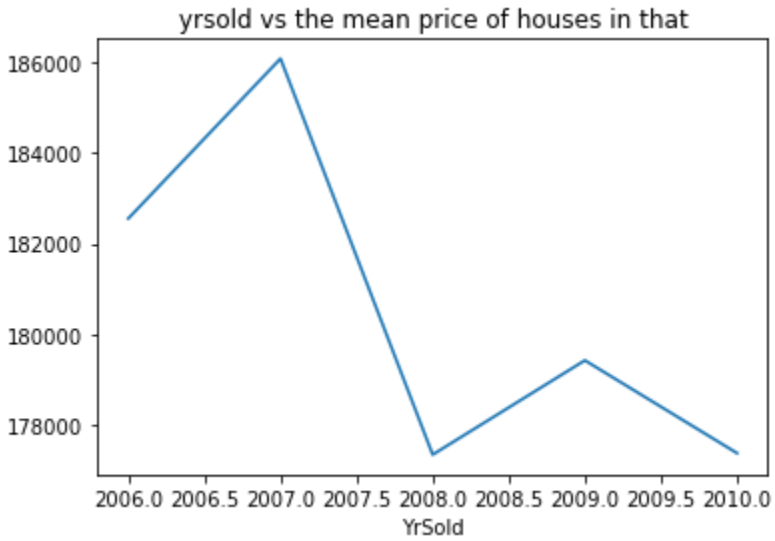
1. EDA - Exploratory Data Analysis:
    a. Looking at what different kinds of features we have
    b. Some feature distributions
    c. And feature relationships with target
2. Dealing with null values
3. Encoding categorical data
4. Scaling data

1.) EDA - Exploratory Data Analysis

- We have the following kinds of features:

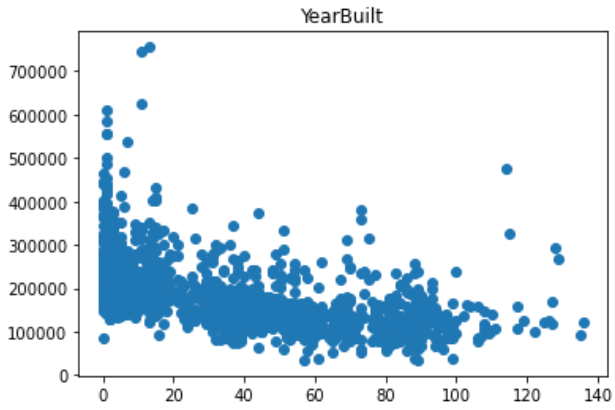
a. Temporal features: time based. Found on the basis of having 'Year' or 'Yr' in their name. *['YearBuilt', 'YearRemodAdd', 'GarageYrBlt', 'YrSold']*
b. Numerical features: Ones which have numerical and non-object data.
c. Discrete feature: Numerical features which have unique values <25. i.e. it has less than 25 classes.
d. Continuous features: Numerical features which are not discrete.
e. Categorical features: Ones which have object data type.

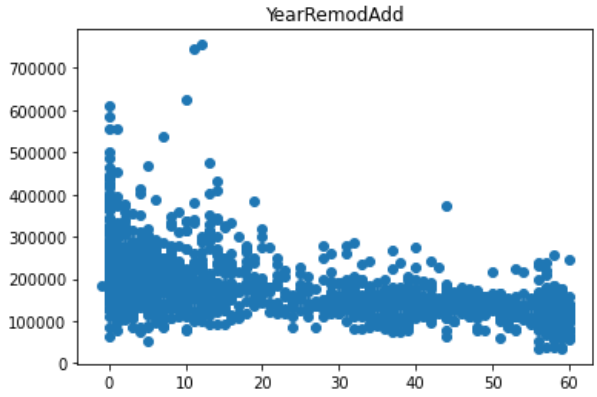
- For the temporal feature: Year sold, we plot a histogram of YearSold vs SalePrice. The following is the plot:

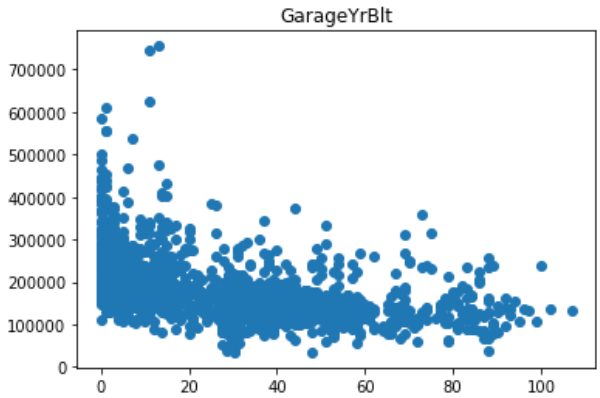


Observation: This is a counter-intuitive plot. Recently sold houses should be more expensive. But what we are missing here is the data about the year of building of house, basement and the year of renovation.

We now plot the graphs of other *temporal_features - YrSold* .







Observation: x-axis has the difference between that feature's year and the year of selling. And the y-axis has SalePrice. We see that for more recently built homes or garages or renovations, the prices are higher.
We conclude that standalone temporal features are useless.

2.) Dealing with Null Values

1. Drop the columns : *'Id'* (index column is not a feature) *,'PoolQC','MiscFeature','Alley','Fence'* (these had null values for >88% of the samples.
2. Some other columns have null values but for <88% of the samples. We impute these.
3. Continuous data is imputed by mean strategy.
4. Discrete data and categorical data (where a null_value is a true noisy null value) is imputed by the mode(most frequent realization).

3.) Encoding Categorical Data

- We have 2 types of categorical data:
    - Numerical-like classes are defined by numbers.
    - String-like classes are defined by string names
- One Hot Encoding of the features is the ideal method as it removes all bias. However doing that for all the categorical variables will increase the total number of features by too much.
- Hence I did label encoding for all the categorical data.

4.) Scaling Data

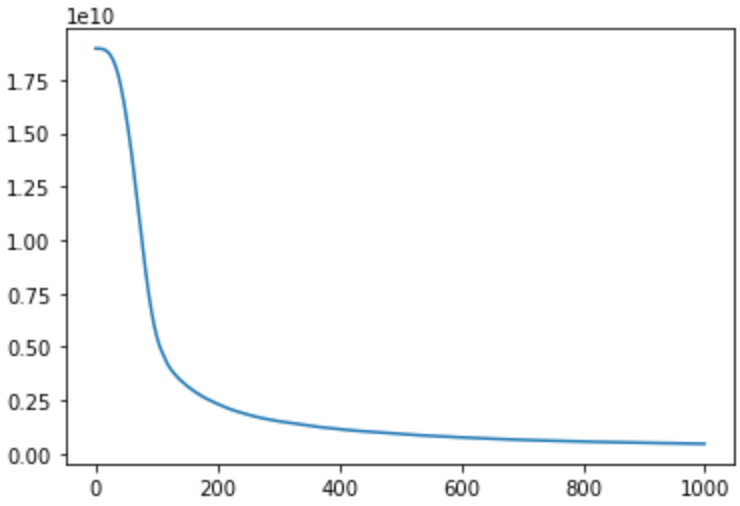
Applied a simple standard scaler on entire dataset as MLP uses gradient descent which needs standard scaling to remove bias between differently ranged features.

Training an MLP:

I used a MLP Regressor with relu activation and 2 hidden layers with sizes -> 150, 100 respectively. This was trained using 70% of the training data. And the prediction was done on the rest of 30%.
(I have not used the test_data provided in kaggle, because for the other questions we have to find MSE and other distance metric loss values for the test data.
This cannot be done, as for the test_data provided we don't have the ground truth labels.)

Result: a net MSE loss of `1188393491.0447512` is reported.

Loss Curve on every step of training:

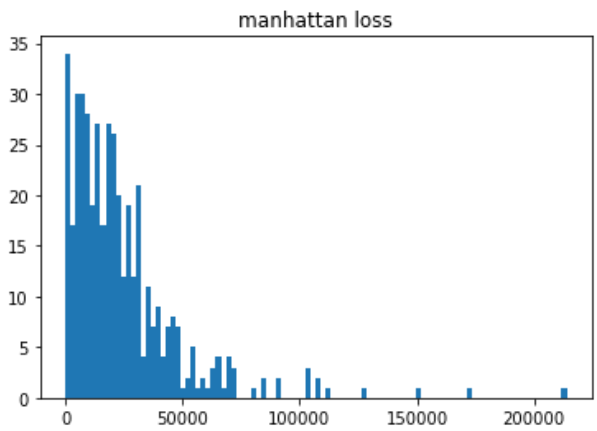


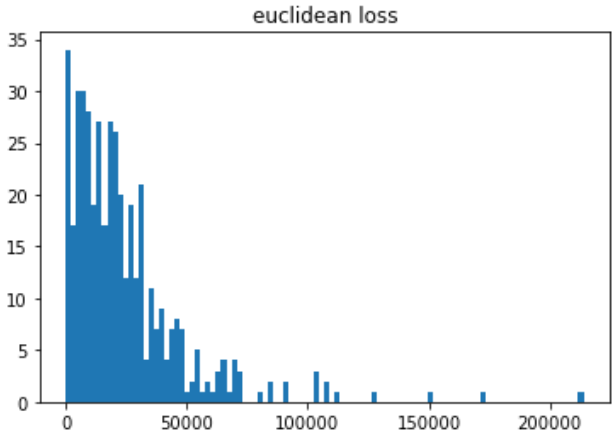
Observation:
On every step of training, the training loss is calculated and plotted. An exponentially decreasing and towards the end converging graph is ideal. Which we were able to achieve.

Evaluating other distance-loss metrics i.e. Euclidean and Manhattan:
Created 4 functions -> for each distance metric two functions. One finds the sample wise loss and the other finds the net loss.
Net loss is reported in comparison to the MSE loss.

```
188393491.0447512 -> MSE net loss
721468.189927734   -> euclidean net loss
10639863.598244008 -> manhattan net loss
```

And the sample wise loss is used to find the distribution of the loss functions.





What?

X-axis has bins of individual sample's Manhattan and euclidean loss. Y-axis has the number of samples that fall in that bin.
Why?

We can see that both the histograms with bins=100 are almost the same. This is because when you have 1D data to compare.. Euclidean and Manhattan metrics perform almost the same.
Apart from the loss curve which proves that out training is proper, these histograms also show that for most of the samples, the euclidean and manhattan losses are near 0.

Link:
https://drive.google.com/file/d/1Za8adEj_0c2eSNHEp4pA8jyyFBNTBuf2/view?usp=sharing

https://drive.google.com/file/d/1TevUqF9Mlh-v0EtLW6CteAGxppLhFaiU/view?usp=sharing