HOUSING PRICES

Kwaku Kessey
Bachelor of Science
Computer Science
University of Rochester
Class of 2020
kkesseya@u.rochester.edu

Muhammad Ahmad
Bachelor of Science
Computer Science
University of Rochester
Class of 2020
mahmad4@u.rochester.edu

ABSTRACT
In this paper, we explore advance regression techniques as well as implementing proper data mining through a final project. The goal of the final project is to develop a data mining solution to predict an outcome, discover patterns or classify events.

Keywords
Regression; Decision Trees; Neural Networks; K-Nearest Neighbor

INTRODUCTION
To demonstrate this group has a good understanding of data mining the project that was chosen was "House Prices: Advanced Regression Techniques", a data mining project competition from Kaggle. Description of project:

> **Problem**: Homebuyers usually have difficulty predicting or estimating the cost of their dream house. They usually have a whole list of attributes and features they want in their house. The problem that occurs is how does a home buyer consolidate or combine all those attributes to determine the price of their dream house.
>
> **Goal:** Using a list of house attributes and features to predict the final sales price of each house.

To solve such a problem two approaches were considered:

> **Naive Approach:** The naive approach to solving the problem listed above is to use a decision tree to determine the cost of houses.
>
> **Advance Approach:** The advanced approach to solving the problem is to use a neural network to determine the cost of the houses.

The general outline of this project is

1. Data Preprocessing
    a. Data Cleaning
    b. Feature Selection
    c. Normalization
2. Applying regression techniques to classify or predict house prices
3. Compare techniques based on a scoring equation provided by Kaggle to see which technique performed the best and why.
4. Explore extra techniques to better understand regression, classification and machine learning.

DATA PREPROCESSING
The data set used for this project was provided by Kaggle it used 79 attributes of a house to provide the sales price of a house. There were a variety of data types among the attributes:

> **Numeric** - takes numeric values on a continuous range
>
> **Nominal** - takes discrete values out of a finite range
>
> **Binary** - specific type of numeric data, only takes value 1 or 0, used to answer yes or no questions
>
> **Ordinal** - is a categorical, statistical data type where the variables have natural, ordered categories and the distances between the categories is not known.

Data Cleaning
The data set consisted of 2800 houses(Tuples). Of these 2800 tuples there were multiple instances were continuous variables(Dimensions) consisted of non-computable values such as "NA." Furthermore, most of the ordinal attributes had string values instead of numeric values, which makes processing the data slightly more complex.

Therefore, for all the instances were "NA" was a value continuous attribute at a tuple it was replaced with 0. There were instances in which 0 couldn't be used to replace "NA", so a different value was used. Below are all the major attributes that had missing values and the replacement process:

> **11 of 80 numeric attributes with "NA" from training set:**
>
> BsmtFullBath, BsmtHalfBath -> 0
>
> No Bathroom in basement
>
> GarageYrBlt = YearBuilt
>
> Garage was built in the same year the house was built
>
> LotFrontage -> 0
>
> MasVnrArea -> 0
>
> BsmtFinSF1, BsmetFinSF2, BsmtUnfSF, TotalBsmtSF -> 0
>
> Basement does not exist
>
> GarageArea, GarageCars -> 0
>
> Garage does not exist

Normalization
Normalization was also a major factor which was considered during preprocessing. Since the numeric values that were in the data set were of such wide ranges, normalization will improve the

processing. The benefits of normalization include: Searching, sorting, and creating indexes fasters, since tables are narrower, and more rows fit on a data page. Therefore, normalization was performed on all possible attributes.

Feature Selection

Another major step during preprocessing is feature selection. Feature selection is a very important step when mining data because it can improve results, increasing processing efficiency, and decrease space usage. Initially when processing the full data set (all 80 attributes) using a decision tree to determine the sales price, overfitting was a major issue. Overfitting is the production of an analysis that corresponds too closely or exactly to a particular set of data and may therefore fail to fit additional data or predict future observations reliably. Therefore, to rectify the issue of overfitting and increase generalization of regression model or algorithm feature selection was used to reduce the amount of attributes considered. We took two approaches to feature selection:

2.3.1 Linear Correlation

Given two attributes, correlation can measure how strongly one attribute implies the other, based on the available data.

For nominal values, we use the chi-square test.

For numeric attributes, we use Pearson's coefficient. We used the following equation to find the correlations:

$$r = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sqrt{\sum (x - \bar{x})^2 \sum (y - \bar{y})^2}}$$

This returns us a value between -1 and 1. If the resulting value is 0, then the two attributes are independent. If the resulting value is less than 0, then the two attributes are negatively correlated, while if the result is greater the 0, they are positively correlated.

Attribute correlations below:

Figure 1 shows correlations of all the 79 attributes with the sales price.

Figure 2, 3 and 4 shows the graph of the twelve highest ranked attributes from the correlation values.

Figure 1:

Ranked attributes:

0.791	OverallQual	0.2145	BsmtUnfSF
0.7086	GrLivArea	0.2096	LotFrontage
0.6404	GarageCars	0.1984	BsmtFinType1
0.6234	GarageArea	0.1807	Exterior1st
0.6136	TotalBsmtSF	0.1759	Exterior2nd
0.6059	1stFlrSF	0.1682	BedroomAbvGr
0.5607	FullBath	0.1656	SaleCondition
0.5339	ExterQual	0.1585	Fence
0.5337	TotRmsAbvGrd	0.1395	Neighborhood
0.5229	YearBuilt	0.1273	HouseStyle
0.5071	YearRemodAdd	0.1242	BldgType
0.4726	MasVnrArea	0.1215	Alley
0.4669	Fireplaces	0.1122	Functional
0.4293	KitchenQual	0.1114	ScreenPorch
0.3921	Foundation	0.1097	BsmtFinType2
0.3864	BsmtFinSF1	0.1089	Condition1
0.3533	BsmtQual	0.1036	RoofMatl
0.3521	FireplaceQu	0.1029	BsmtCond
0.339	HeatingQC	0.1001	ExterCond
0.3335	GarageFinish	0.0935	PoolQC
0.3244	WoodDeckSF	0.0924	PoolArea
0.3223	GarageType	0.0914	Heating
0.3193	2ndFlrSF	0.072	MiscFeature
0.3159	OpenPorchSF	0.0682	LotConfig
0.3117	MasVnrType	0.0505	LandSlope
0.2841	HalfBath	0.0464	MoSold
0.2672	GarageCond	0.0446	3SsnPorch
0.2638	LotArea	0.041	Street
0.2614	GarageYrBlt	0.0343	LandContour
0.2513	CentralAir	0.0308	14 Condition2
0.2474	LotShape	0.0143	Utilities
0.2434	SaleType	-0.0114	BsmtFinSF2
0.2417	MSZoning	-0.0168	BsmtHalfBath
0.2384	GarageQual	-0.0212	MiscVal
0.236	Electrical	-0.0256	LowQualFinSF
0.2283	BsmtExposure	-0.0289	YrSold
0.2278	PavedDrive	-0.0779	OverallCond
0.2271	BsmtFullBath	-0.0843	MSSubClass
0.2223	RoofStyle	-0.1286	EnclosedPorch
		-0.1359	KitchenAbvGr

Figure 2:

The following figure shows the correlation graphs of OverallQual, GrLivArea, GarageCars and GarageArea. All of these attributes have a positive correlation of at least 0.6 with SalesPrice. A reason for the correlation to be so high is that almost none of these have any outliers. One can argue that GarageCars and Garage Area might have outliers, but they are not as prominent to affect the training data.

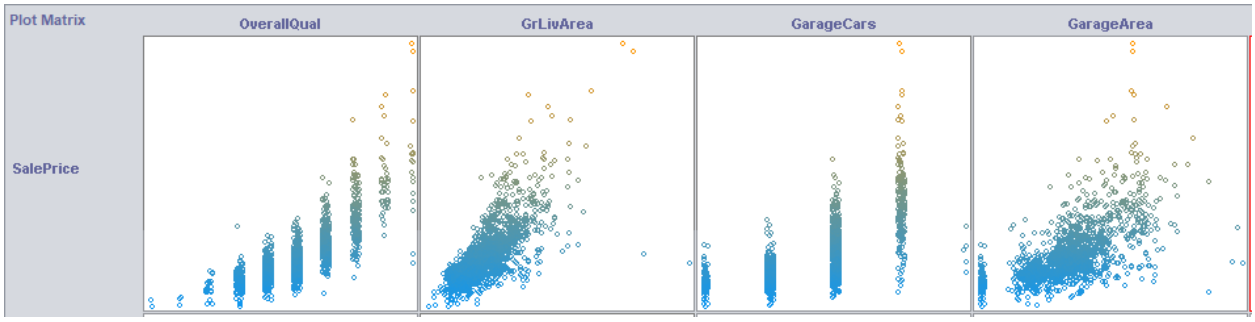


Figure 3:

The following figure shows the correlation graphs of TotalBsmtSF, 1stFlrSF, FullBath and TotRmsAbvGrd with SalesPrice. All of these attributes have a positive correlation with SalesPrice. It is important to note TotalBsmtSF and 1stFlrSF each have one outlier.

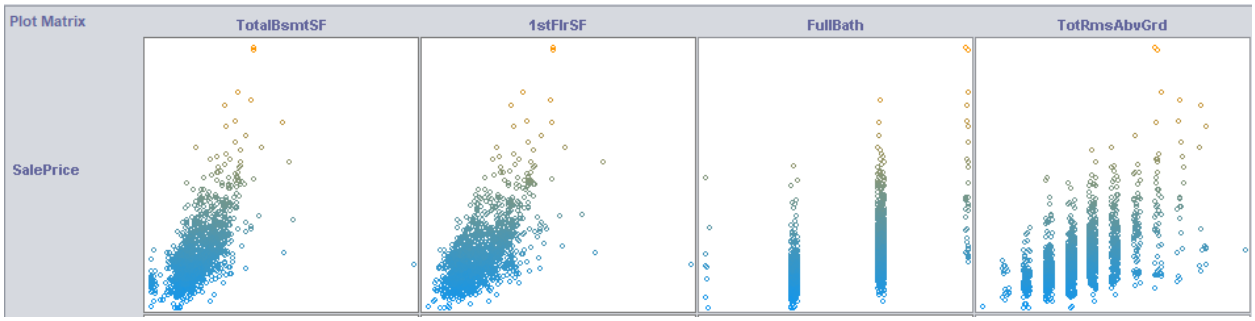


Figure 4:

The following figure shows the correlation graphs of YearBuilt, YearRemodAdd, MasVnrArea and Fireplaces. All of these have a positive correlation with SalesPrice with only a few outliers.

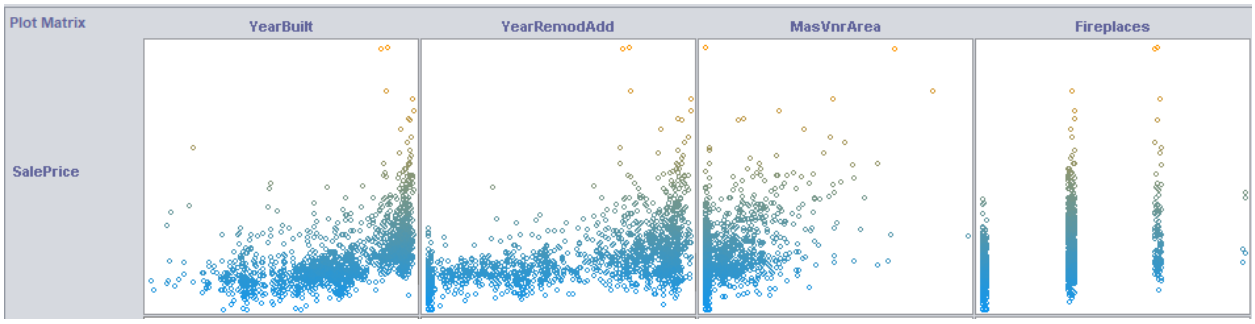


Figure 5:

It is a scatter-plot matrix of MSSubClass, OverQual, GrLivArea and TotRmsAbvGrd

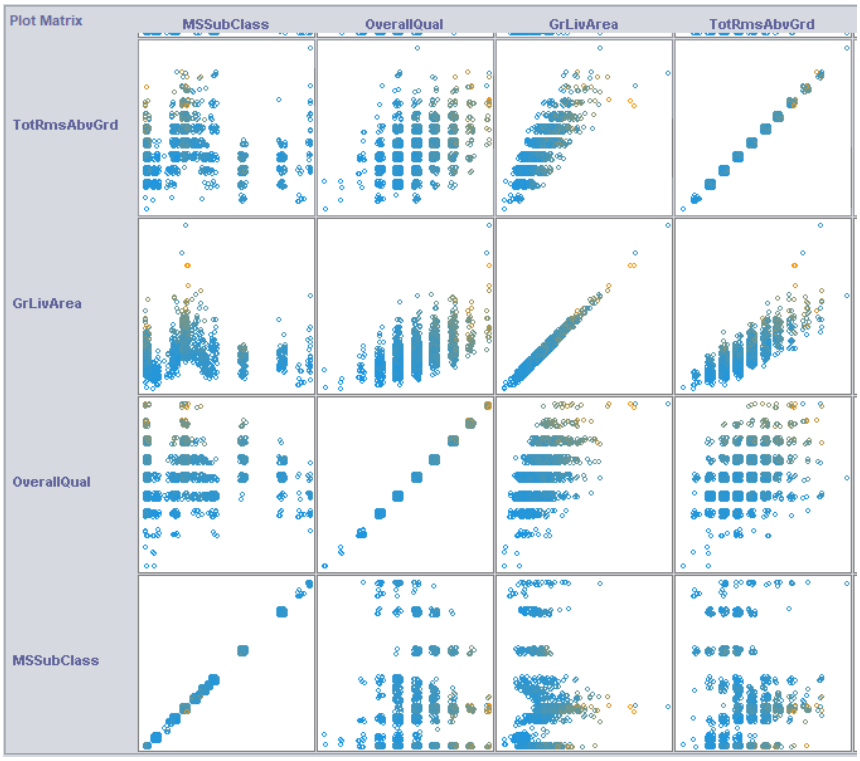
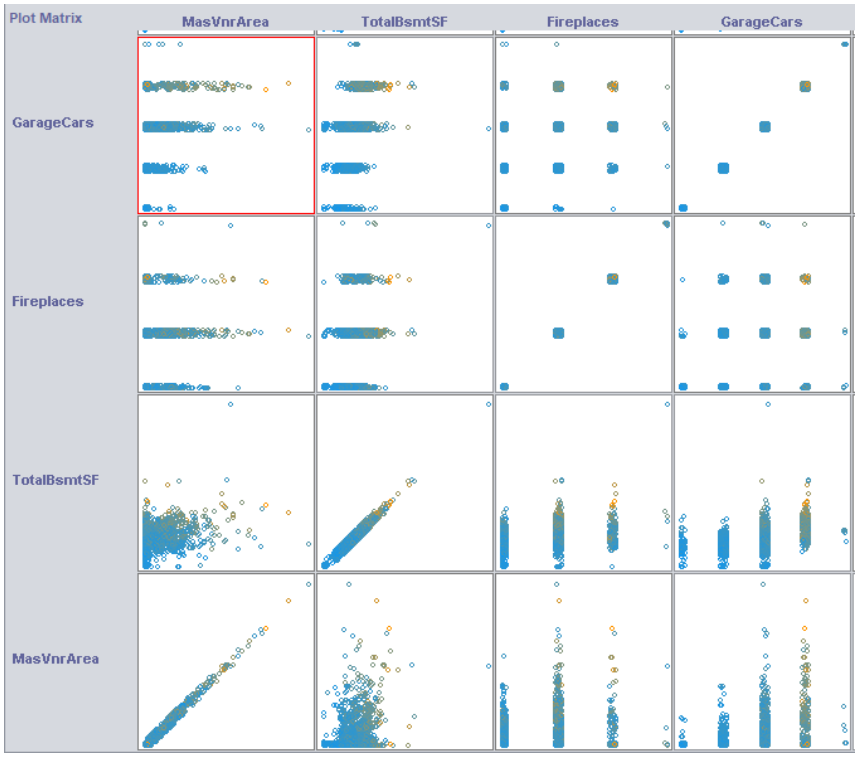
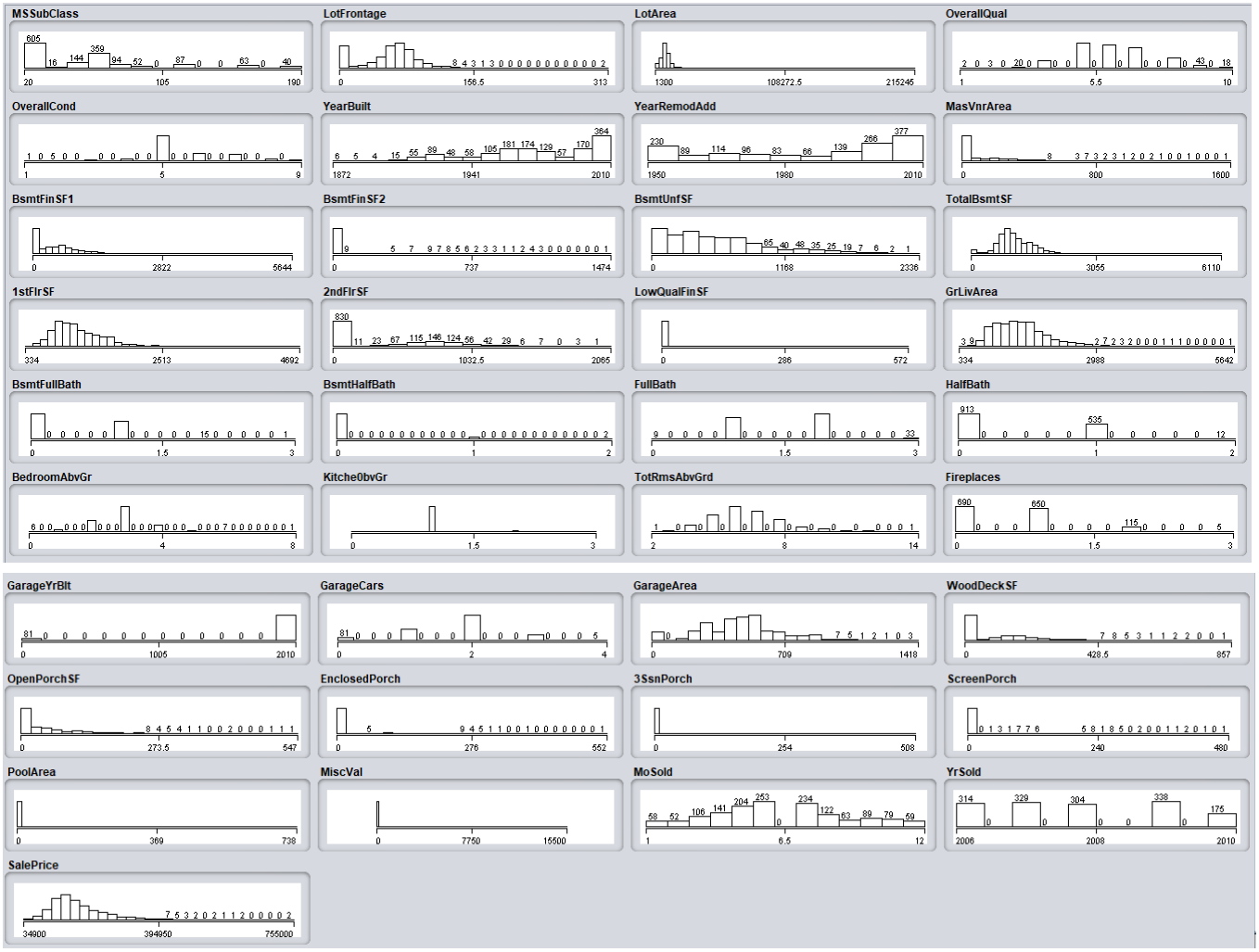


Figure 6:

It is a scatter-plot matrix of MasVnrArea, TotalBsmtSF, Fireplaces and GarageCars



The following graphs show the values and the frequency of the reduced attributes after feature selection(described below):



2.3.2 Feature Scoring

This feature selection technique is a part of the Java Machine Learning Library and rank attributes based on gain ratios.

Out of those two approaches feature scoring from the Java ML library proved most effective in producing accurate results for all the variety of attributes. In all out of the 80 attributes 43 attributes were removed because they had zero correlation on the sales price, all the other attributes had a correlation.

1. How Will the Data Be Evaluated?

For the evaluation of our results from the techniques in section 4 the Root-Mean-Square-Error was used:

$$Score = \sqrt{\frac{\sum_{i=1}^{n}(\log \hat{Y}_i - \log Y_i)^2}{n}}$$

The log price is to reduce the impact of biased higher price. Therefore, the most optimal result or score from each technique will be zero and the worst possible score will be infinity.

TECHNIQUES

To perform the processing and sales price prediction we used a regular decision tree, and Neat Neural Network. In addition to the two approaches K-Nearest-Neighbor, Decision Tree K-Nearest-

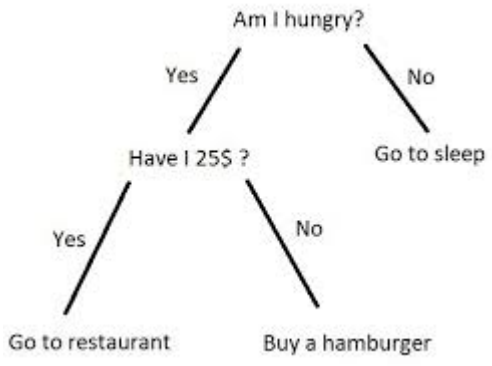
Neighbor, and Random Forest Regression were also tested for extra credit.

1.1 Decision Tree

The first naïve technique we used was a Decision Tree. A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

"Decision Tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each node holds a class label. The top most node in the tree is the root node." (Han, 330)

The following is an example of a decision tree:



Gini index was used to measure the impurity, a data partition or set of training tuples, of the attributes at each stage. The most 'pure' attribute was then chosen to split the tuples.

Equation of Gini index:

$$Gini(split) = \sum_{i=1}^{n} p_i * (1 - p_i).$$

Results: With the basic decision tree the evaluation score was

0.3213579352013241

K-Nearest Neighbors – Classification

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique. (Sayad, n.d.)

KNN classifies classes based on distances. There are two ways it calculates distances one way for continuous values and the other for nominal values. As seen below:

Continuous Value Distance Calculation:

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

Manhattan $\sum_{i=1}^{k}|x_i - y_i|$

Minkowski $\left(\sum_{i=1}^{k}(|x_i - y_i|)^q\right)^{1/q}$

Nominal Value Distance Calculation:

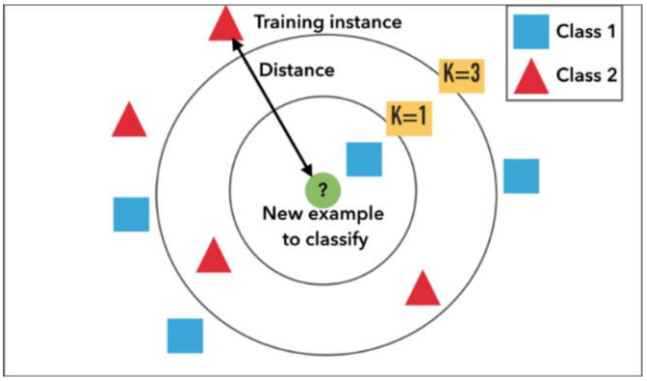
Hamming Distance

$$D_H = \sum_{i=1}^{k}|x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

The distances are then used to classify. For example:



Example of k-NN classification. The test sample (inside circle) should be classified either to the first class of blue squares or to the second class of

red triangles. If k = 3 (outside circle) it is assigned to the second class because there are 2 triangles and only 1 square inside the inner circle. If, for example k = 5 it is assigned to the first class (3 squares vs. 2 triangles outside the outer circle). (Bronshtein, 2017)

For the project we used a KNN classifier that uses 5 neighbors to decide the sales price.

Results: With the K-Nearest Neighbor Classifier the evaluation score was

0.35984575625531534

K-Dimensional Tree K-Nearest Neighbor

The difference between a traditional decision tree as in section 4.1 and this decision tree is the K-Nearest Neighbor. The benefit of using K-Nearest Neighbor with decision tree is that it should make the algorithm faster, at the cost of memory and more time spend during building or training the classifier.

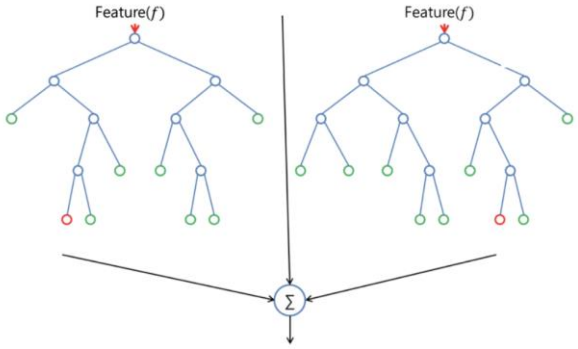
The configuration for the processing or training was using 5 neighbors to make decisions.

Results:

0.35984575625531534

Random Forest Classification

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. (Wikepedia, n.d.) For example, multiple trees will be constructed as seen below and then combined to form a generalized good fit decision tree.



When combining decision trees as seen above in random forest a weighting algorithm is used. For Java ML, the machine learning algorithm library used in this project, it does not detail or state the exact algorithm it uses. However, generally after training, predictions for unseen samples x′ can be made by averaging the predictions from all the individual regression trees on x′. Algorithm:

$$\hat{f} = \frac{1}{B}\sum_{b=1}^{B} \hat{f}_b(x')$$

A valuable benefit of random decision forests is that it can correct for decision trees' habit of overfitting to their training set. Overfitting is a major issue with conventional decision trees, but with the method above from random forests it reduces that by creating multiple random decision trees.

For the project we used a random forest with a tree count of 50 and no our of bag error estimates. To train the classifier a reduced attribute set with 1460 tuples were used, and to test the classifier a test set of 1460 tuple were used.

Results:

0.35984575625531534

Neuroevolution Of Augmenting Topologies Neural Network

NeuroEvolution of Augmenting Topologies (NEAT) is a genetic algorithm (GA) for the generation of evolving artificial neural networks (a neuroevolution technique) developed by Ken Stanley in 2002 while at The University of Texas at Austin. It alters both the weighting parameters and structures of networks, attempting to find a balance between the fitness of evolved solutions and their diversity. It is based on applying three key techniques: tracking genes with history markers to allow crossover among topologies, applying speciation (the evolution of species) to preserve innovations, and developing topologies incrementally from simple initial structures ("complexifying"). (Wikepedia, Neuroevolution of augmenting topologies, n.d.)

A difference between NEAT and Feed Forward Neural Network is that a NEAT NN can back track and cycle its network. In addition, in a NEAT NN can generate multiple layers to assist its prediction.

The NEAT Neural network program used for this program is from the Encog Machine Learning Framework. A k-fold cross validation of 5 was used when training the neural network.

Result:

0.2105181235075658

Feed Forward Neural Network

A feedforward neural network is an artificial neural network wherein connections between the nodes do not form a cycle.

The feedforward neural network was the first and simplest type of artificial neural network devised. In this network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes and to the output nodes. The network is based on neurons which form this mathematical equation seen below from the book “Artificial Intelligence A Modern Approach”:

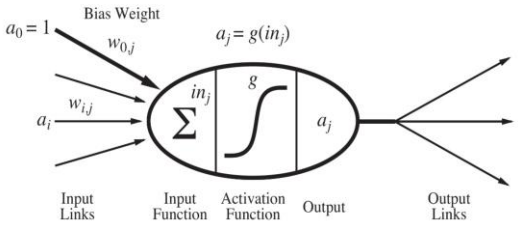
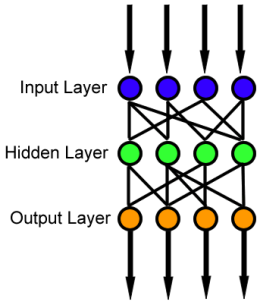


Figure 18.19 A simple mathematical model for a neuron. The unit's output act
$a_j = g(\sum_{i=0}^{n} w_{i,j} a_i)$, where a_i is the output activation of unit i and $w_{i,j}$ is the wei
link from unit i to this unit.

In the case of this project there is only one hidden layer within the feed forward neural network, which is the sales price. For example, in the image below we see an input layer, which in our case are the reduced attributes, a hidden layer, and an output layer.



The configuration for the processing or training was using a k-fold cross validation of 5, to reduce overfitting.

Results:

0.2292024497634854

COMPARE RESULTS

We implemented 5 different Machine Learning algorithms. We started with a naïve decision tree and from there went on to advance algorithms. For each algorithm we evaluated our scores using the equation from section 3. The best possible solution for each technique mention in section 4 would have been 0 and the worst would have been infinity. For all the techniques we implemented we got a score of better than 0.5.

Our first technique, decision tree, described in the section 4.1 returned us a score of 0.32. This was possible because a decision tree can assign specific values to problem, decisions and outcomes of each decision; however, we were afraid that it may be overfitting the data. Therefore, we decided to implement K-Nearest Neighbors-Classification, described in section 4.2, and K-Dimensional Tree K-Nearest Neighbor. Both of these techniques did equally well, with a score of 0.359, but did not give us a score better than the basic decision tree.

With the second and the third technique failing to provide us with a better score, we decided to implement Random Forest Classification as it would correct the decision tree's overfitting and may return us a better score. However, this was not the case as it returned us a score of 0.3598, lower than the decision tree. This maybe because it fails to account for rare outcomes.

With all the techniques scoring below the decision tree. We decided to use a Feed Forward Neural Network, described in section 4.6. Feed Forward Neural Network can represent more complex functions easily; therefore, this was the perfect technique to use on a data set with 80 attributes. This technique returned us a score of 0.229, which was considerably better than the score returned by the Decision Tree.

The Neat Neural network, described in section 4.5, returned us an even better score of 0.2105

CONCLUSION

Initially we ran the decision tree, which gave us a threshold score. Most often the naïve technique may work better than some of the complex ones. We tried a few other techniques explained above but were not able to get better results as compared to the decision tree. We were determined to get a better score than the decision tree's, as it was a naïve approach and we thought it would be running into a case of overfitting. The Neat Neural Network returned us the best score!

REFERENCES

Han, Jiawei. Data Mining: Concepts and Techniques. Morgan Kaufmann Publishers.