Thomas Power

February 28th, 2023

Note: Enclosed in this folder are report, corresponding python files, and dataset. If you have any questions please do not hesitate to reach out. Thank you!

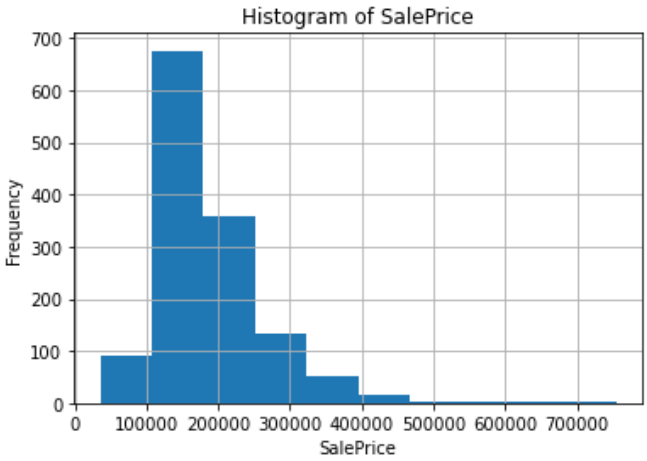
Dataset:
For this project I wanted to explore one of my passions which is real estate. I was able to locate a dataset on Kaggle, https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data that contained data about homes in Ames, Iowa. This dataset contained two files

a training dataset and a test dataset. The test dataset had a missing class so for this assignment I strictly used the training dataset and split this up accordingly. In this data there are 81 attributes. They include items such as square footage, bathrooms, bedrooms, condition, etc. These attributes are all details about the property.

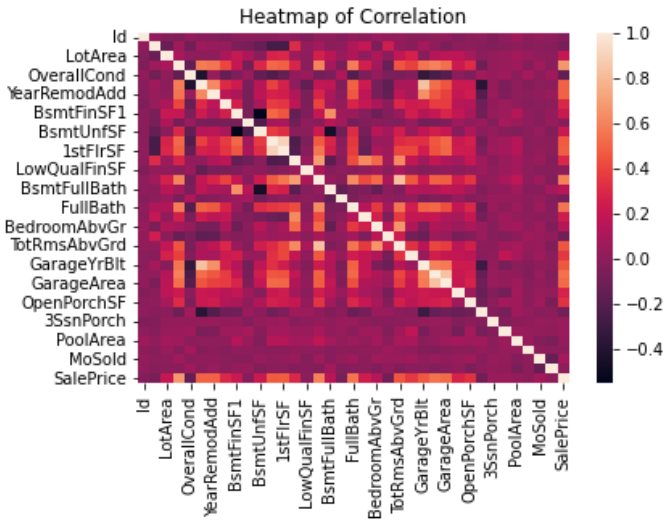
Goal:
The goal of this project is to use this dataset to determine if a house is "expensive" or "affordable". Expensive is defined as anything over the mean sale price, while affordable is anything equal to or below the mean sale price.

Data Preprocessing:
The first step completed was to visually inspect the data. In doing so, I noticed that there was a significant amount of NAs present in the file. Some columns, such as *Alley,* had almost all NAs. I decided to further explore this and see the percentage of NAs present in each column. From there, I eliminated any column that had more than 15% NAs. In doing so I eliminated the following six columns: *LotFrontage, Alley, FireplaceQu, PoolQC, Fenc, MiscFeature*. At this point, I was left with 1460 rows and 75 columns. Due to the number of missing data still present, I decided to remove the remaining NAs rows. This left me with 1338 rows after processing.

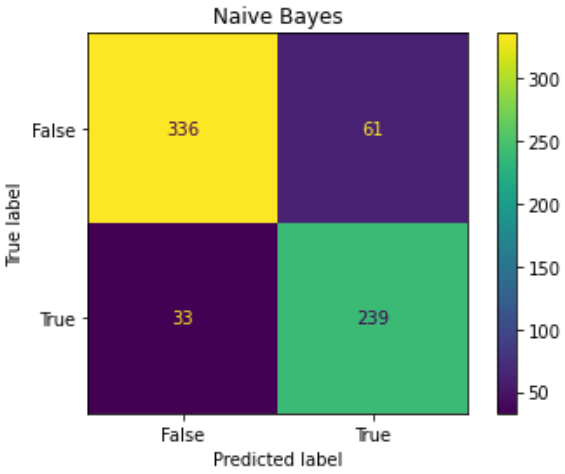
Picking Class Label
There were a few additional steps that I took before moving onto the classification methods. First, I took a look at the SalePrice. I decided to plot this as a histogram, pictured to the right. I wanted to make sure it was somewhat regularly distributed and there were no extreme outliers. From this graph we can see that the data falls within a range we would expect for a house, ranging from about $50,000 to $750,000. The data appear to be somewhat normal with a high frequency around $150,000. Next, I took the mean, median, and mode of the sales price. The mean is $186,761.78, the median is $168,500.00, and the mode is $140,000. I wanted to pick this as my class label, so I split SalePrice into two sets. Anything greater than the mean of $186,761 will be considered 1 or "expensive" and anything less than or equal to $186,761 will be considered 0 or "affordable".



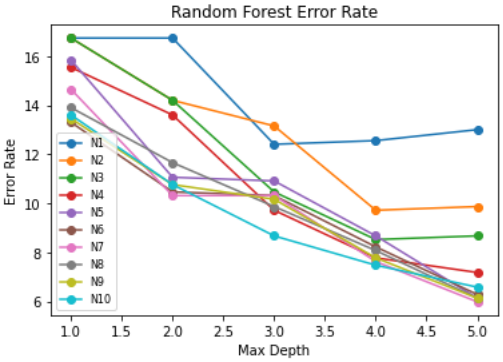
Correlation:
Next, I wanted to look at the correlation. Which attributes are having the biggest impact on the SalePrice? First, I plotted this in the figure below. I also listed out the values of the correlation. I



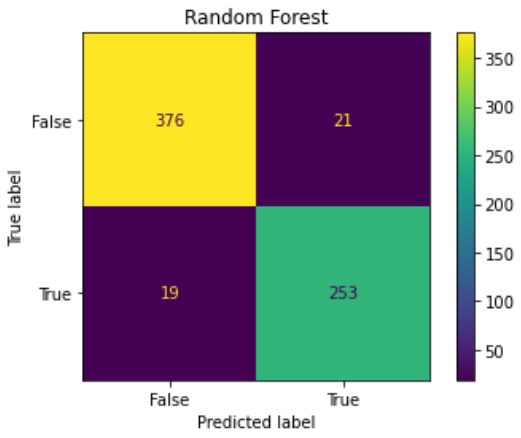
found that the attributes with the highest positive correlation to SalePrice are OverallQual, GrLivArea, FullBath, GarageCars, GarageArea, YearRemodAdd, YearBuilt, TotalBsmtSF, GarageYrBlt, and TotRmsAbvGrd. Based on my previous knowledge, this correlation makes sense to me. I would think attributes like Overall Quality, Gross Living Area, number of bathrooms, etc. would have a large impact on the SalePrice. I made a subset of these attributes and focused on them for the remainder of my project. I will run all my classifiers on these and compare accuracies. I decided to go with a 50/50 training testing split. Lastly, I applied StandardScaler to my data before running my classifiers in order to increase accuracy.



Naïve Bayes:
The first algorithm I ran was Naïve Bayes. This is an unsupervised learning method. In this algorithm we are assuming independence between variables. We then are calculating the probability of an event occurring. For example what is the probability that the house is affordable if the year built was 1972, square footage 2,000 square feet, etc? We then pick the class that it has a higher probability of belonging to. On the right you can see the correlation matrix. We can see we have 239 true positives and 336 true negatives. In this classifier I got an accuracy of 85.949%. This means that 85% of the time Naïve Bayes is correctly picking the right SalePrice class.



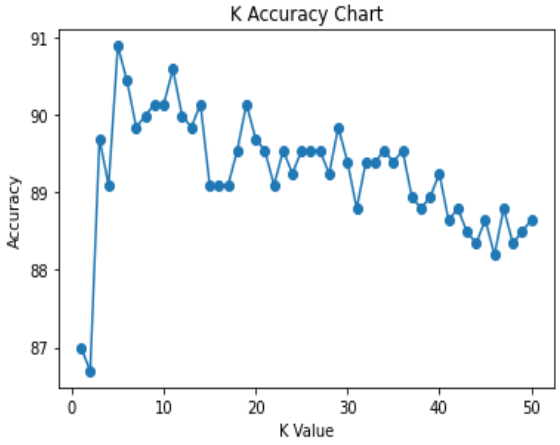
Random Forest:
The second algorithm I ran was Random Forest. I picked this algorithm because it is another supervised learning method but different than naïve bayes. I wanted to see how they stack up against each other. Here, we are forming many trees in order to make our prediction. Then, we took the most frequent prediction result and used this to predict. Here I decided to try different configurations for

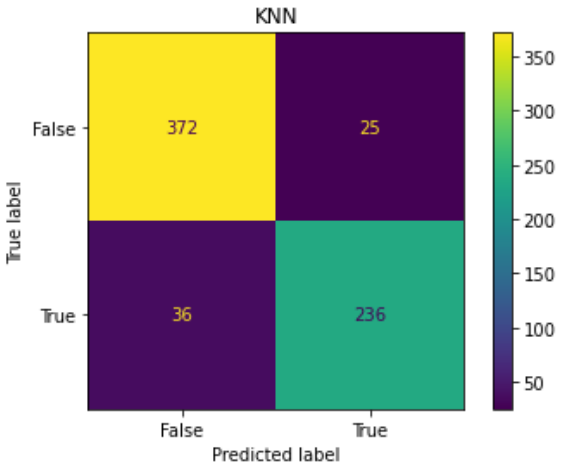


the max depth and the number of n. I tried n values 1-10 with max depth 1-5. When doing so, I kept track of the error rate as we are trying to minimize this. Above to the right is a plot of the error rates. From this I was able to determine the best combination to minimize the error rate is an n value of 7 and max depth of 5, since this is the lowest point of the graph. Above to the left is the confusion matrix. There are 253 true positives and 376 true negatives. With these configurations I got an accuracy of 94.02%. So 94% of the time we are selecting the correct SalePrice class. This is about a 10% improvement from Naïve Bayes.

KNN:

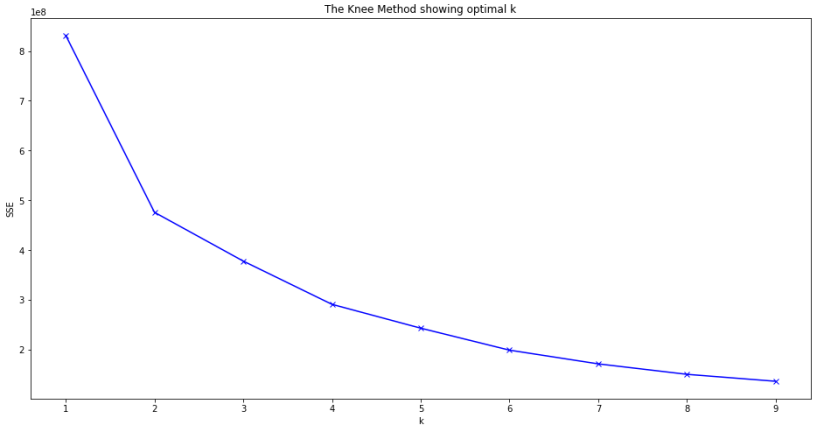
The last supervised learning method I ran was the KNN algorithm. Here we are determining distance that each point is from the neighbor. We are then taking the smallest distance and using this to make our prediction. I tried k values from 1-50. I plotted the accuracy in a diagram, located below. From this diagram we can see the highest accuracy is with a k value of 5, since this is the highest point on the graph. A confusion matrix is located below. There are 236 true positives and 372 true negatives. For my problem this configuration yields an accuracy of 90.88%. This is slightly lower than Random Forest, but still about 5% higher than Naïve Bayes.



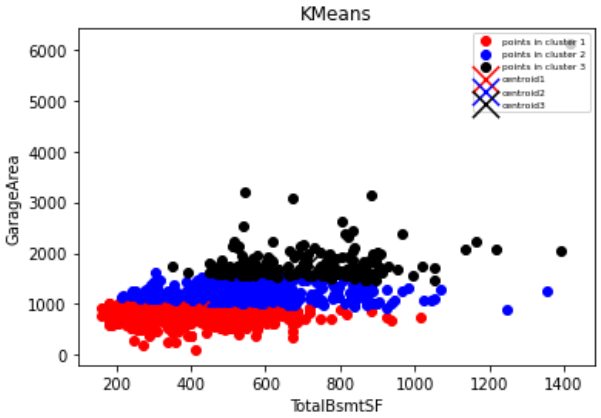


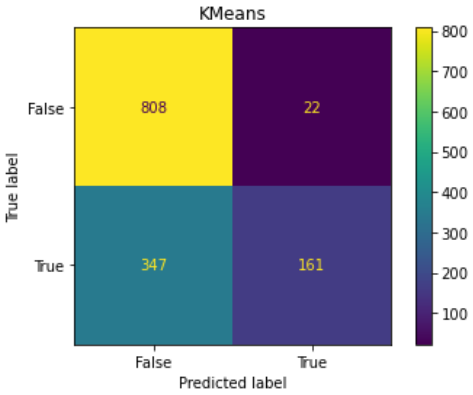
Kmeans:

Lastly, I ran Kmeans algorithm. I wanted to see how an unsupervised classification method compared to my supervised models. In this algorithm we are splitting the data into clusters in order to classify. I started out by plotting the k values from 1 to 10, pictured below. Here we are using the knee method to determine the best value for k. From this chart I am going to pick a value of 3. After 3 the chart begins to level off, and the tradeoff for adding another k is not worth it. Then I decided to randomly select two columns. In my case were GarageArea', 'TotalBsmtSF'. I then plotted these in a chart located below. I then assigned labels to each cluster based on the majority. Cluster 1 and 2



had a majority of class 0, while cluster 3 had a majority of class 1. Then I went through every point in the dataset to see which centroid it was closest to and assigned it this class. A confusion matrix is located below as well. There are 161 true positives and 808 true negatives. Lastly, I calculated the accuracy which was 74.42%. This is lower than all three of the supervised methods I used previously by about 10-20%. This model is correctly classify about 74% to the correct SalePrice class.





Conclusion:

Below is a summary of my results. From this table we can see that Random Forest has the highest TPR while Kmeans has the lowest. We can see that Kmeans has the highest TNR while Naïve Bayes is the lowest. From an accuracy standpoint, kmeans is the lowest and Random Forest is the highest. Random forest has an accuracy rate of 94%. I decided to pick accuracy as the most important factor. The number of true positives and TN are very important in this case. This would mean that the model is predicting the correct class. In conclusion, I will take Random Forest as my top performer. Random Forest with n value of 7 and max depth of 5 has an accuracy 94% meaning 94% of the time it can predict if a property in Ames, Iowa is expensive or affordable.

Model	TP	FP	TN	FN	Accuracy	TPR	TNR
Naïve Bayes	239	61	336	33	85.9491779	0.87867647	0.84634761
Random Forest	253	21	376	19	94.0209268	0.93014706	0.94710327
KNN	236	25	372	36	90.8819133	0.86764706	0.93702771
Kmeans	161	22	808	347	72.4215247	0.31692913	0.97349398

How to run code:

I broke my code into four files: naive_bayes.py, random_forest.py, knn.py, and kmeans.py. Each runs the classifier that is in the name. The only thing to note is that naïve_bayes.py includes some of the initial charts such as the histogram and correlation plot.

Citations:

House prices - advanced regression techniques. Kaggle. (n.d.). Retrieved December 15, 2022, from https://www.kaggle.com/competitions/house-prices-advanced-regression-techniques/data

Khan, M. (2017, August 2). *KMEANS clustering for classification*. Medium. Retrieved December 15, 2022, from https://towardsdatascience.com/kmeans-clustering-for-classification-74b992405d0a

Random forest classifier: A complete guide to how it works in Machine Learning. Built In. (n.d.). Retrieved December 15, 2022, from https://builtin.com/data-science/random-forest-algorithm

Sagar, R. (2021, February 18). *What is a naive Bayes classifier and what significance does it have in ML*. Analytics India Magazine. Retrieved December 15, 2022, from https://analyticsindiamag.com/what-is-a-naive-bayes-classifier-and-what-significance-does-it-have-for-ml/

What is the K-nearest neighbors algorithm? IBM. (n.d.). Retrieved December 15, 2022, from https://www.ibm.com/topics/knn