Module 4 Assignment: Nashville Housing

College of Professional Studies, Northeastern University

Professor: Justin Grosz

Nov 4, 2022.

Name: Karan Gigoo

NUID: 001581360

**Introduction**

In this assignment we have the data for the different factors that are affecting the price of the properties in the City of Nashville Tennessee. In the data we have 22651 rows and 26 different columns that give us information regarding how the different factors are affecting the prices of the houses. Factors such as Full Bath, Land Value, Acreage, etc. are few to name. Our main task is to understand how these factors are affecting the price and how this would allow us to invest in the required areas.

**Data cleaning**

The data cleaning is one of the most important phases before we begin the modeling of our data. Keeping that in consideration, one of most important factors that influence the model inverse are null values and this set of data has a few. Text

Description automatically generated

Figure 1 Null values

Here we see that the column ‘Suite/Condo’ is completely null so we can drop this column as it has no information that is useful to us. Apart from this we also observe that columns such as ‘Half Bath’, ‘Full Bath’, ‘Bedrooms’, ‘Foundation Type’, ‘Finished Area’, ‘Property City’, and ‘Property Address’ have null values. As the number of null values for these are not as significant in number as the ‘Suite/Condo’ there is much alteration that is necessary. Hence, the rows that how this data can be dropped and help in the cleaning of the data.

**EDA**

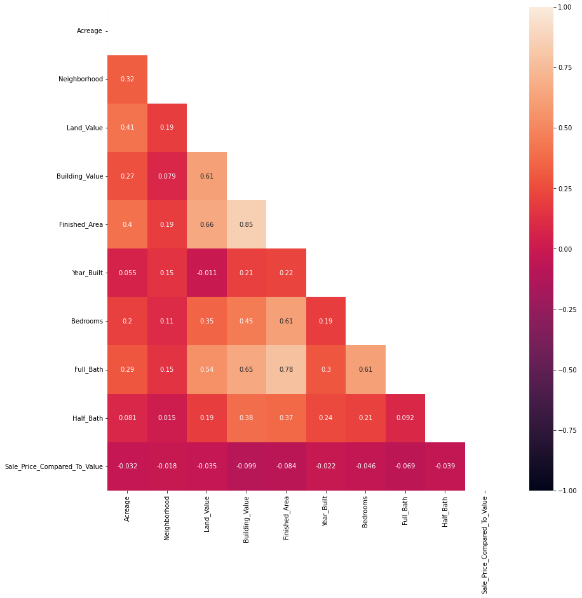
In this section we talk about the exploratory data analysis that allows us to get a better understanding of how the will trend. This will allow us to better understand what the different points are the data is trying to tell us and how this interacts with the working of the models. We drop the columns ‘Unnamed:0’, ‘Parcel\_ID’, ‘Sale\_Date’, ‘Property\_Address’. We also drop columns such as ‘Property\_City’, ‘City’, ‘Legal\_Reference’, ‘State’ as we know the city, we are targeting is Nashville and the state is Tennessee, the other columns are something that would create a large and inefficient model hence they are dropped . The reason for this drop is that these factors do not hold any significance when considering the price of the property and worsen the model when considered.

Figure 2 Correlation Map

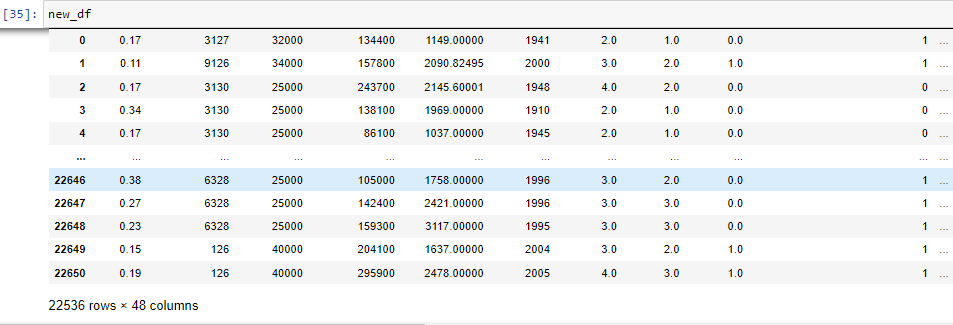
To get a better understanding of how the data is related to the target we create correlation plot that allows us to get a better understanding of how the data is trending and what are the hidden relation in the data. As we can see in the correlation plot most of the data is weakly correlated to our target Variable. Most of the data is categorical in nature. This means that we will have to encode these values before we feed them to the model. Also, encoding allows us to get a detailed understanding of how each individual category is affecting the target value. After encoding the data, we get the following data frame.

Figure 3 After Encoding

**Analysis**

**Modeling:**

For the modeling process we are using the model phase we are using multiple classification techniques that will allow us to see which technique fits the best. First, we use logistic regression as it works well with binary outputs.

**Logistic regression:**

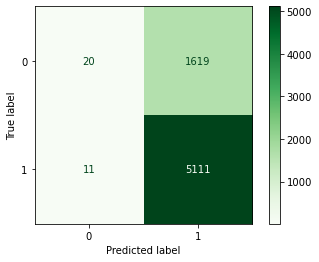
Logistic regression is a technique where the target variable is in the form of a binary which means it is either 1 or 0. For the set we have our target where we are checking if the property is overpriced or underpriced. The model has an accuracy of 0.75891. With a precision of 0.75943 and the recall of 0.99785. With this information we get the following confusion matrix. Here, we have 20 true positives, 5111 False Negatives, 1619 False Negatives and 11 True Negatives.

Figure 4 Confusion Matrix Logistic

As we can see, we are trying to see if the target model predicting for the price of the more than the actual price hence the number of False Negative are high. The False positives are where the prices are predicted to be low, but the price should be higher. The number of True negatives is low and are the house that have been predicted to have a higher price but are under values and these are the ones the investors should be careful of. Since the prediction in the confusion matrix are not up to mark, we move to other models.

**Decision Tree:**

Decision Tree helps us in understanding the model better and gives us a better understanding compare the variables better. Our decision tree has a depth of 4 which means there would be 4 instances of cross validation. For the model we get an accuracy of 0.76497. We get a precision of 0.76 for 0 and 0.77 for 1. We get a recall of 0.04 for 0 and 1.00 for 1. Table

Description automatically generated Here, we see that there are 76 True Positives, 1556 True Negatives, 23 False Positives and 5099 False Negatives.

Figure 5 Matrix Decision Tree

The for the model we have better precision 0.77 compared to the previous model. The accuracy of the model is also better than the previous model hence it is better at predicting. The number of True negatives is low and are the house that have been predicted to have a higher price but are under values and these are the ones the investors should be careful of . The output is better than the previous model.

For the features that are driving the prediction of the model, Chart

Description automatically generated

Figure 6 Feature Selection

Here we see the features that drive the prediction of model are whether the house is Vacant, the value of the building and the Land Value. These features are important as the value of the building and the price of the land hold key in checking if the building is overpriced or underpriced. The house being not vacant is important as an occupied will have people living on the premises have an affect on the on others buying in the house.

**Random Forest:**

Unlike decision trees where the trees are created individually, in random forest all the branches are created simultaneously hence it is more effective. The depth of is model is 4 which means that there would be 4 cross validation points that would be created for the model. Table

Description automatically generated

Figure 7 Matrix Random Forest

In our matrix we get an accuracy of the model as 0.76676 . The precision of our model is 0.85 for 0 and 0.76 for 1. A recall we get 0.02 0 and 1.00 for 1. Here, we see that there are 33 True Positives, 1606 True Negatives, 6 False Positives and 5116 False Negatives. The for the model we have better precision 0.85 compared to the previous model. The accuracy of the model is also better than the previous model hence it is better at predicting. The number of True negatives is low and are the house that have been predicted to have a higher price but are under values and these are the ones the investors should be careful of . The output is better than the previous model.

The features for the model we have

Chart

Description automatically generated

Figure 8 Features Random Forest

Here we see that the features such as the vacancy of the house, building value, Finishes Area, and the Land Value hold key. These features are important as the value of the building, the land area, and the price of the land hold key in checking if the building is overpriced or underpriced. The house being not vacant is important as an occupied will have people living on the premises influence the on others buying in the house. Unoccupied and vacant homes often pose a greater risk of damage, vacant home insurance is expensive.

**Gradient Boosting :**

In gradient boosting, it utilizes a method called boosting that combines weak learners sequentially, so that each new tree corrects the errors of the previous one. Table

Description automatically generated

Figure 9 Gradient Boosting Matrix

In the above matrix we get model accuracy of 0.76260 where the precision of our model is 0.58 for 0 and 0.77 for 1. A recall we get 0.07 0 and 0.98 for 1. Here, we see that there are 111 True Positives, 1528 True Negatives, 77 False Positives and 5045 False Negatives. Compared to the previous model we have lower precision and lower accuracy than the previous model. The number of True negatives is low and are the house that have been predicted to have a higher price but are under values and these are the ones the investors should be careful of . The output isn’t better than the previous model.

For the features driving the model

**Chart

Description automatically generated**

Figure 10 Features Gradient Boost

Here we see that the building value, land value , the year the house is built in, the finished area, and the neighborhood the house belongs are the top features that are affecting the model. These features are important as the value of the building, the land area, and the price of the land hold key in checking if the building is overpriced or underpriced. Here, the neighborhood in which the house belongs also affects the prices as an area with amenities would have an overall higher cost.

**Benchmarking Metrics :**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr No. | Accuracy | Precision | Recall |
| Logistics | 0.75891 | 0.75943 | 0.99785 |
| Decision Tree | 0.76497 | 0.774122 | 1.00 |
| Random Forest | 0.76676 | 0.851639 | 1.00 |
| Gradient Boosting | 0.76260 | 0.775122 | 0.985122 |

Benchmarking metrics allows us to get a better understanding of which model to select based on the precision, accuracy, and the recall of the models. Here, we see that **Random Forest** provides the best solution.

**Conclusion**

To conclude I would say that factors such as Land Value, Building Value and the vacancy of the house are the factors that are driving all the model and help us in predicting how the factors are affecting the price. The land Value and the Building Value are key factors as they give us a fair estimate of how the overprice of the house is estimate . The vacancy of the house is also a key factor as it allows us to understand how different external factor that aren’t mentioned influence a particular variable(Insurance of the vacant home, the amenities of the house if not vacant).

With help of the bench marking metrics, we can understand that over iteration of all the models, **Random Forest** has the highest accuracy and the least False negative. Here, the reduction of the false negatives is key are we don’t want the investment by the company to put on values that being predicted to be high but are underpriced.

A recommendation to the company would be to use the **Random Forest** Algorithm as it provides the best solution amongst the selected models.

**Reference:**

* Timmons, M. (2021, December 15). When to get unoccupied and vacant Home Insurance. ValuePenguin. Retrieved December 4, 2022, from <https://www.valuepenguin.com/unoccupied-and-vacant-home-insurance#:~:text=You%20should%20be%20prepared%20to,annual%20cost%20of%20homeowners%20insurance>.
* Home. MindTools. (n.d.). Retrieved December 4, 2022, from <https://www.mindtools.com/az0q9po/decision-tree-analysis>
* By: IBM Cloud Education. (n.d.). What is Random Forest? IBM. Retrieved December 4, 2022, from <https://www.ibm.com/cloud/learn/random-forest#:~:text=Random%20forest%20is%20a%20commonly,both%20classification%20and%20regression%20problems>.
* Dhingra, C. (2020, December 28). A visual guide to gradient boosted trees. Medium. Retrieved December 4, 2022, from <https://towardsdatascience.com/a-visual-guide-to-gradient-boosted-trees-8d9ed578b33>

**Appendix 1**

**Code:**

#!/usr/bin/env python

# coding: utf-8

# In[5]:

import pandas as pd

from sklearn.model\_selection import train\_test\_split

import statsmodels.api as sm

from statsmodels.stats.outliers\_influence import variance\_inflation\_factor

from sklearn import tree

from sklearn.tree import DecisionTreeClassifier

import matplotlib.pyplot as plt

from sklearn.linear\_model import LogisticRegression

from sklearn import metrics

from sklearn.metrics import accuracy\_score

import seaborn as sns

import numpy as np

from matplotlib import pyplot

from sklearn.metrics import classification\_report, confusion\_matrix,plot\_confusion\_matrix

# In[6]:

data = pd.read\_csv("C:/Users/kgrat/OneDrive/Documents/ALY 6020/week 4 - Nashville\_housing\_data.csv")

# In[7]:

data.shape

# In[8]:

data.head(10)

# In[9]:

data.columns = ['Unnamed: 0', 'Parcel\_ID', 'Land\_Use', 'Property\_Address',

'Suite/Condo', 'Property\_City', 'Sale\_Date', 'Legal\_Reference',

'Sold\_As\_Vacant', 'Multiple\_Parcels\_Involved\_in\_Sale', 'City', 'State',

'Acreage', 'Tax\_District', 'Neighborhood', 'Land\_Value',

'Building\_Value', 'Finished\_Area', 'Foundation\_Type', 'Year\_Built',

'Exterior\_Wall', 'Grade', 'Bedrooms', 'Full\_Bath', 'Half\_Bath',

'Sale\_Price\_Compared\_To\_Value']

# In[10]:

data.info()

# In[11]:

data.isna().sum()

# In[12]:

data.isnull().sum()

# In[13]:

data =data.drop('Suite/Condo',axis = 1)

# In[14]:

df = pd.DataFrame(data)

# In[15]:

df.isna().sum()

# In[16]:

df['Full\_Bath'].value\_counts()

# In[17]:

df['Sale\_Price\_Compared\_To\_Value'].value\_counts()

# In[18]:

df.isna().sum()

# In[19]:

df = df.dropna(how = 'any')

# In[20]:

df.isna().sum()

# In[21]:

df['Sale\_Price\_Compared\_To\_Value']=df['Sale\_Price\_Compared\_To\_Value'].replace("Under",0).replace("Over",1)

# In[22]:

df = df.drop(columns=(['Unnamed: 0','Parcel\_ID','Sale\_Date','Property\_Address']),axis =1)

# In[25]:

c = df.corr()

# In[26]:

mask = np.triu(np.ones\_like(df.corr(), dtype=bool))

# In[27]:

plt.figure(figsize=((14,14)))

sns.heatmap(c,annot=True, mask=mask, vmin=-1, vmax=1)

# In[28]:

df['State'].value\_counts()

# In[29]:

df[df['Property\_City'].str.match('NASHVILLE')]

# In[30]:

df = df.drop(columns=['Property\_City','City','Legal\_Reference','State'], axis = 1)

# In[31]:

df

# In[32]:

df['Sale\_Price\_Compared\_To\_Value']=df['Sale\_Price\_Compared\_To\_Value'].replace("Under",0).replace("Over",1)

# In[33]:

df['Sale\_Price\_Compared\_To\_Value'].value\_counts()

# In[34]:

new\_df = pd.get\_dummies(df,columns=['Land\_Use','Sold\_As\_Vacant','Multiple\_Parcels\_Involved\_in\_Sale','Tax\_District','Exterior\_Wall','Grade','Foundation\_Type'])

# In[35]:

new\_df

# In[36]:

y=new\_df['Sale\_Price\_Compared\_To\_Value']

x=new\_df.drop('Sale\_Price\_Compared\_To\_Value',axis=1)

# In[37]:

y.shape

# In[38]:

x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=.30, random\_state=123)

# ### Logistics Regression

# In[39]:

logistic\_reg = LogisticRegression()

model=logistic\_reg.fit(x\_train,y\_train)

# In[40]:

pred= model.predict(x\_test)

print('Accuracy of the model is:{:.5f}'.format(model.score(x\_test, y\_test)))

# In[41]:

print("Precision:",metrics.precision\_score(y\_test,pred))

print("Recall:",metrics.recall\_score(y\_test,pred))

# In[42]:

plot\_confusion\_matrix(logistic\_reg,x\_test, y\_test,cmap='Greens')

plt.show()

# ### Decison Tree

# In[43]:

dt\_model = DecisionTreeClassifier(criterion="gini", random\_state=42,max\_depth=4, min\_samples\_leaf=5)

dt\_model.fit(x\_train,y\_train)

# In[44]:

dt\_pred = dt\_model.predict(x\_test)

# In[45]:

acc\_score = accuracy\_score(y\_test,dt\_pred)

# In[46]:

acc\_score

# In[47]:

from sklearn.metrics import classification\_report, confusion\_matrix

print(confusion\_matrix(y\_test, dt\_pred))

print(classification\_report(y\_test, dt\_pred))

# In[48]:

importances = dt\_model.feature\_importances\_

indices = np.argsort(importances)

fig, ax = plt.subplots(figsize=[5, 4])

ax.barh(range(len(importances)), importances[indices])

ax.set\_yticks(range(len(importances)))

\_ = ax.set\_yticklabels(np.array(x\_train.columns)[indices])

ax.set(xlim=(0, 2),ylim=(30, 50))

# ### Random Forest

# In[49]:

from sklearn.ensemble import RandomForestClassifier

rf\_model = RandomForestClassifier(random\_state=42,max\_depth=4)

rf\_model.fit(x\_train,y\_train)

# In[50]:

rf\_pred = rf\_model.predict(x\_test)

# In[51]:

accuracy\_score(y\_test,rf\_pred)

# In[52]:

print(confusion\_matrix(y\_test, rf\_pred))

print(classification\_report(y\_test, rf\_pred))

# In[53]:

importances = rf\_model.feature\_importances\_

indices = np.argsort(importances)

fig, ax = plt.subplots()

ax.barh(range(len(importances)), importances[indices])

ax.set\_yticks(range(len(importances)))

\_ = ax.set\_yticklabels(np.array(x\_train.columns)[indices])

ax.set(xlim=(0, 2),ylim=(30, 50))

### Gradient Boosting

# In[54]:

from sklearn.ensemble import GradientBoostingClassifier

gb\_model = GradientBoostingClassifier(random\_state=42,max\_depth=4)

gb\_model.fit(x\_train,y\_train)

# In[55]:

gb\_pred = gb\_model.predict(x\_test)

accuracy\_score(y\_test,gb\_pred)

# In[56]:

print(confusion\_matrix(y\_test, gb\_pred))

print(classification\_report(y\_test, gb\_pred))

# In[57]:

importances = gb\_model.feature\_importances\_

indices = np.argsort(importances)

fig, ax = plt.subplots()

ax.barh(range(len(importances)), importances[indices])

ax.set\_yticks(range(len(importances)))

\_ = ax.set\_yticklabels(np.array(x\_train.columns)[indices])

ax.set(xlim=(0, 2),ylim=(30, 50))