Prediction of Home's Current Market Value

Zhen Li

I used python to construct the model to predict the value of the home in the market.

Panda, numpy, matplotlob, sklearn, xgboost, tensorflow and keras were utilized to analyze the dataset.

Codes:

```
import pandas as pd
import numpy as np
import matplotlib as mpl
mpl.use('TkAgg')
import matplotlib.pyplot as plt
from datetime import datetime
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
import xgboost as xgb
from sklearn.metrics import r2_score
```

Data Preprocessing

First, I drop the **UseCode** because all home in training and test are single family house. I also observed that **censusblockgroup** can be dropped because median values of each block group are listed in the file. What's more, **PropertyID** is only a unique ID of home and won't contribute to the price. Thus, these three attributes can be dropped to facilitate the training process.

Next, I simply check the null entries in the dataset:

Codes:

```
hw = pd.read_csv('Data Science ZExercise_TRAINING_CONFIDENTIAL1.csv')
print hw.isnull().sum()
```

Output:

GarageSquareFeet	2841
ViewType	8956
BGMedHomeValue	6
BGMedRent	2631
BGMedYearBuilt	247

Only a few entries of **BGMedHomeValue** and **BGMedYearBuilt** are null. Thus, I simply replace the null with their median values.

Codes:

```
temp = hw['BGMedYearBuilt'].median()
hw['BGMedYearBuilt'].fillna(temp,inplace = True)
temp = hw['BGMedHomeValue'].median()
hw['BGMedHomeValue'].fillna(temp,inplace = True)
```

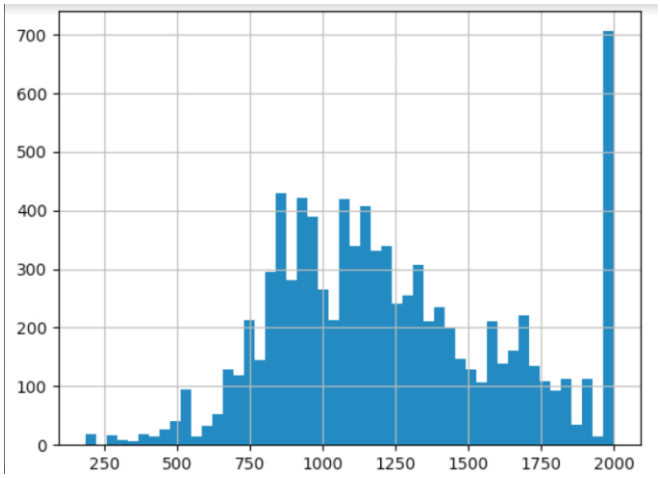
Description of **GarageSquareFeet**: the size of protected garage space if any. So, I replace the null with 0 because null means there is no protected garage space.

Codes:

```
hw['GarageSquareFeet'].fillna(0,inplace = True)
```

However, a large proportion of **BGMedRent** is null. If I simply replace them with any values, the overall distribution of rent would be disturbed substantially. Also, **BGMedRent** may have a strong relationship with the selling price so that I can't drop it. Therefore, the **BGMedRent** was transformed into categorical feature, and all null entries were replaced by "unknown" label.

Distribution of BGMedRent



Judging from the distribution, the **BGMedRent** was split into 5 parts: [0, 1000], (1000, 1250], (1250, 1600], (1600, infinity) and "unknown". I split data in this way because I found that the contribution of [0, 1000], (1250, 1600] and (1600, infinity) were very small when I tuned the parameters during training process. And the count of each feature was almost the same (around 2000) to prevent overfitting.

Codes:

```
def bin_column(col, bins, labels, na_label='unknown'):
  hw[col] = pd.cut(hw[col], bins=bins, labels=labels, include_lowest=True)
  hw[col] = hw[col].astype('str')
  hw[col].fillna(na_label, inplace=True)

bin_column('BGMedRent', bins=[0,1000,1250, 1600, max(hw.BGMedRent)], labels=['0-1', '1-1.25', '1.25-1.6', '>'], na_label='unknown')
```

The **ViewType** was also transformed into categorical feature. The labels are: 1, 2, 3, 4, 5, 6, 7 and "unknown".

Codes:

```
bin_column('ViewType', bins=[0,78,79, 82,241,244,246,247], labels=['1', '2', '3', '4', '5', '6', '7'], na_label='unknown')
```

TransDate is the date of current sale. I converted it into a measure of the number of days from transaction date to 10/27/2019.

Codes:

```
hw.TransDate = pd.to_datetime(hw.TransDate)
hw['days'] = (datetime(2019,10,27)-hw.TransDate).astype('timedelta64[D]')
```

ZoneCodeCounty is the intensity of use or density the lot is legally allowed to be built-up to. Only those features with count >80 were kept and all other features were replaced by "other" label.

Codes:

```
Zlist = ['SF 5000','R6','R4','RA5','R5','SF 7200','R8','SR6','RS7200','R3.5','RS7.2','RSA 6','R1', 'RA2.5','MU','R6P','R7','RSX 7.2','RS9.6','URPSO','R9.6','RS 8.5','UL7200','LDR','SR4.5']
hw.loc[~hw.ZoneCodeCounty.isin(Zlist),'ZoneCodeCounty'] = 'other'
```

All values in **Longitude** were transformed to a positive number by adding 3.6*10^8 because of the log transformation in the next step.

Machine Learning

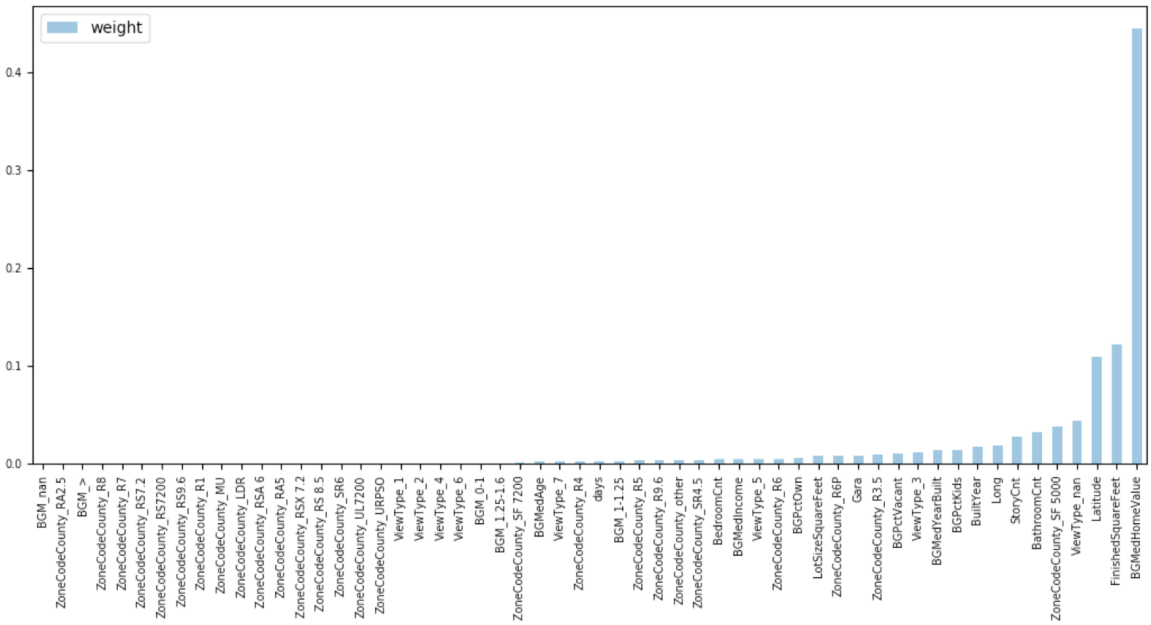
The categorical variables were one-hot encoded using get_dummies() from pandas, the numerical variables went through a log transformation and were standardized using StandardScaler() from sklearn. A train-test split was performed with a test size of 0.1.

I used XGBoost methods because XGBoost is likely to provide the best achievable accuracy as observed in my research practice. The evaluation metrics were r Squared Value, Average Absolute Percent Error (AAPE) and Median Absolute Percent Error (MAPE).

Codes and results:

```
xtrain,xtest,ytrain,ytest = train_test_split(x,y,test_size=0.1,random_state=1500)
# Fitting the model
reg = xgb.XGBRegressor()
reg.fit(xtrain, ytrain)
training_preds = xgb_reg.predict(xtrain)
val_preds= xgb_reg.predict(xtest)
ytrain = np.exp(ytrain)
training_preds = np.exp(training_preds)
ytest = np.exp(ytest)
```

```
val_preds = np.exp(val_preds)
# Printing the results
print("Training r2:", round(r2_score(ytrain, training_preds),4))
print("Validation r2:", round(r2_score(ytest, val_preds),4))
print("Training AAPE:", round(mean_absolute_percentage_error(ytrain, training_preds),4))
print("Validation AAPE:", round(mean_absolute_percentage_error(ytest, val_preds),4))
print("Training MAPE:", round(median_absolute_percentage_error(ytrain, training_preds),4))
print("Validation MAPE:", round(median_absolute_percentage_error(ytest, val_preds),4))
```



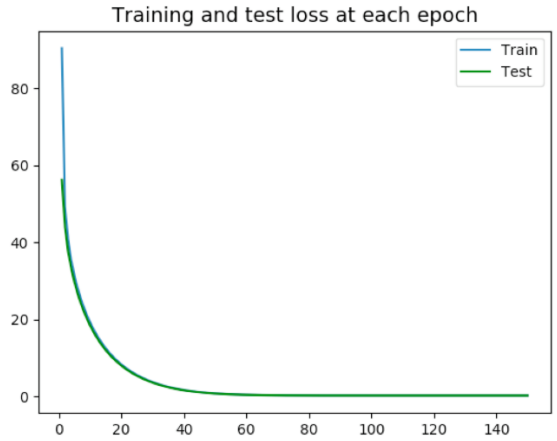

```
('Training r2:', 0.8468)
('Validation r2:', 0.8479)
('Training AAPE:', 12.3698)
('Validation AAPE:', 12.9836)
('Training MAPE:', 8.8194)
('Validation MAPE:', 9.3575)
```

These results are pretty good for a machine learning model without tuning the hyperparameters. From the weight plot we can find that the most important feature is the BGMedHomeValue. Although it seems that half of features have zero importance, they may still have interactions with other important features.

Besides XGBoost, I also tried neural networks to see if I could improve the performance. I used a 4-layer densely connected neural networks with L1 adams regularization. However, the AAPE and MAPE of the neural network were much worse than the results from XGBoost model.

Codes and Results:

```
from keras import models, layers, regularizers
# Building the model
nn = models.Sequential()
nn.add(layers.Dense(128, input_shape=(xtrain.shape[1],), kernel_regularizer=
regularizers.l1(0.005),  activation='relu'))
nn.add(layers.Dense(256,kernel_regularizer= regularizers.l1(0.005), activation='relu'))
nn.add(layers.Dense(256,kernel_regularizer= regularizers.l1(0.005), activation='relu'))
nn.add(layers.Dense(512,kernel_regularizer= regularizers.l1(0.005), activation='relu'))
nn.add(layers.Dense(1, activation='linear'))
# Compiling the model
nn.compile(loss='mean_squared_error', optimizer='adam', metrics=['mean_squared_error'])
# Training the model
nn_history = nn.fit(xtrain,ytrain,epochs=150,batch_size=256,validation_split = 0.1)
```




```
('Training r2:', 0.7266)
('Validation r2:', 0.6381)
('Training AAPE:', 66.3366)
('Validation AAPE:', 64.0161)
('Training MAPE:', 45.2072)
('Validation MAPE:', 43.7433)
```

Conclusion

Judging from the experiment results, I decide to predict the **SaleDollarCnt** using the XGBoost model I developed from training dataset. The AAPE is estimated to be around 12.9 and MAPE is estimated to be around 9.3 based on the cross-validation results. Please see the attached csv file for reference.

To further improve the performance of the machine learning, we can tune the hyperparameters during each epoch. Also, we can study the multi-collinearity of all features. It's obvious that there are strong colinear relationship between the median values in each block group. A heatmap can help us figure out the relationship and drop the collinear features.