**UTSAV PATEL**

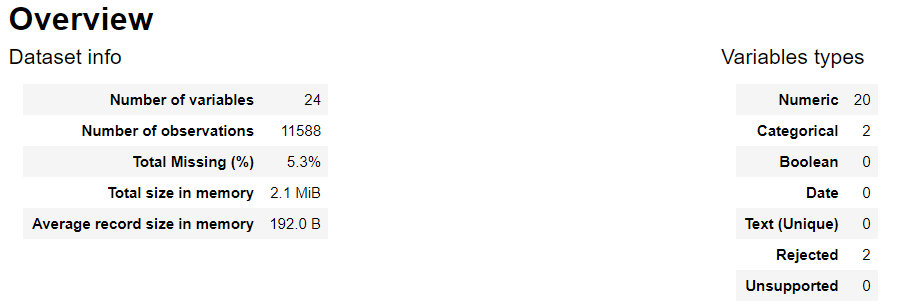
Indiana University-Bloomington

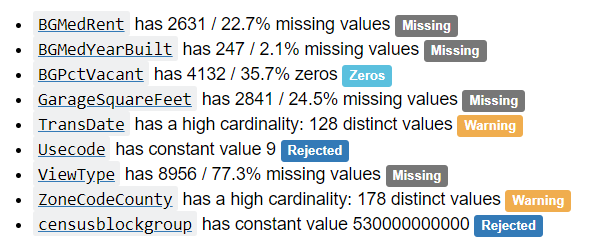
2661, E 7th Street, Apt. E, Bloomington,IN-47408

utpatel@iu.edu | [utsavpatel3797@gmail.com](mailto:utsavpatel3797@gmail.com) | 312- 774-3172

**Zillow Project Report:**

* **Exploratory Data Analysis from pandas\_profiling library:**

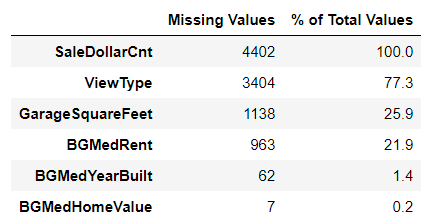
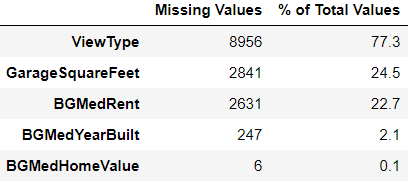




**Data Preprocessing :**

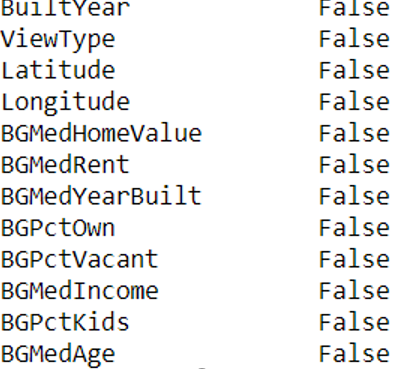
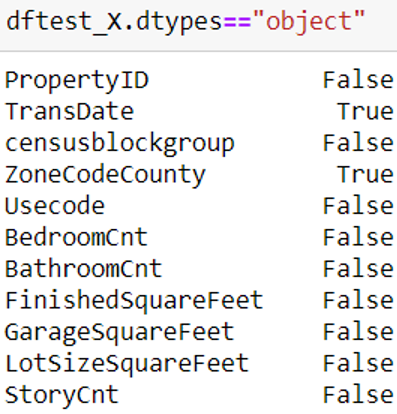
* **Checking Missing Values:**

**Train Missing Values :** **Test Missing Values:**



Note here that in above test missing values, SaleDollarCnt will obviously be missing in the test file, so no interpretation should be done from it.

* **Check categorical variables:**



Thus, there are only 2 categorical variables: TransDate and ZoneCodeCounty

* **Treating categorical Variables**

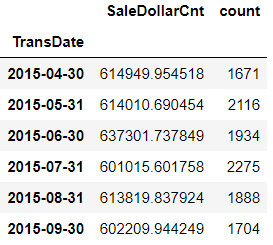
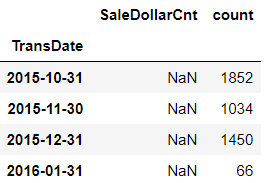
1. **TransDate:**

**How I treated TransDate and Why I dropped TransDate Column:**

First approach that came to my mind was to create bins based on months and that lead

to below counts:

Train File: Test File:

While in test we had different months. Also, there was no specific pattern that we could infer from above figure. So, I thought of creating new features based on day of transaction (Sunday, Monday, .. ). But, again no useful pattern was found in that approach. So, I decided to **drop TransDate** Column.

1. **ZoneCodeCounty :**

Clearly One Hot Encoding was best approach for this variable.

ZoneCodeCounty Unique Values in **Train : 178**

ZoneCodeCounty Unique Values in **Test : 143**

After One Hot Encoding of Train and Test file:

**Total Train Columns: 199**

**Total Test Columns : 164**

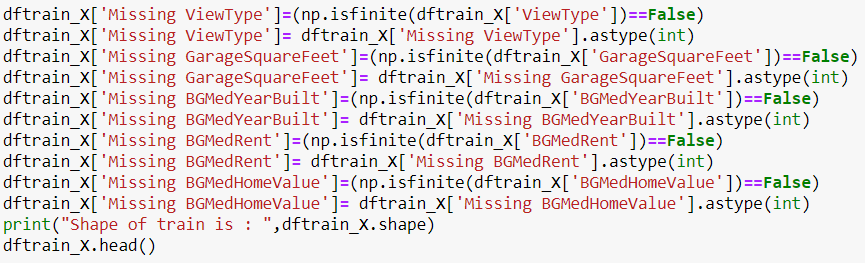
In order to keep train and test files in same dimensions, I fetched columns which were in testfile and not in trainfile and created new columns in trainfile. After that I dropped train columns which were not in test file. Now, at the end of this step, **we have 164 rows in both train and test**.

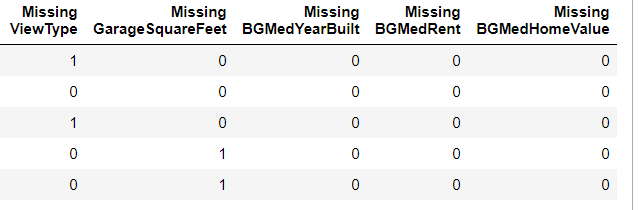
1. **“Censusblockgroup” and “Usecode” – Dropped :**

Censusblockgroup and Usecode were same in both the files(train and test). Thus, I **dropped it**. Now there are 162 columns in train and test.

* **Creating new Columns:**

New columns were created from columns which had missing values. If there is missing values in a column, the corresponding data will have 1 if data is missing and 0 if data is not missing.





* **Scaling the Data :**

Used sklearn.preprocessing.StandardScaler() to scale the dataframe.

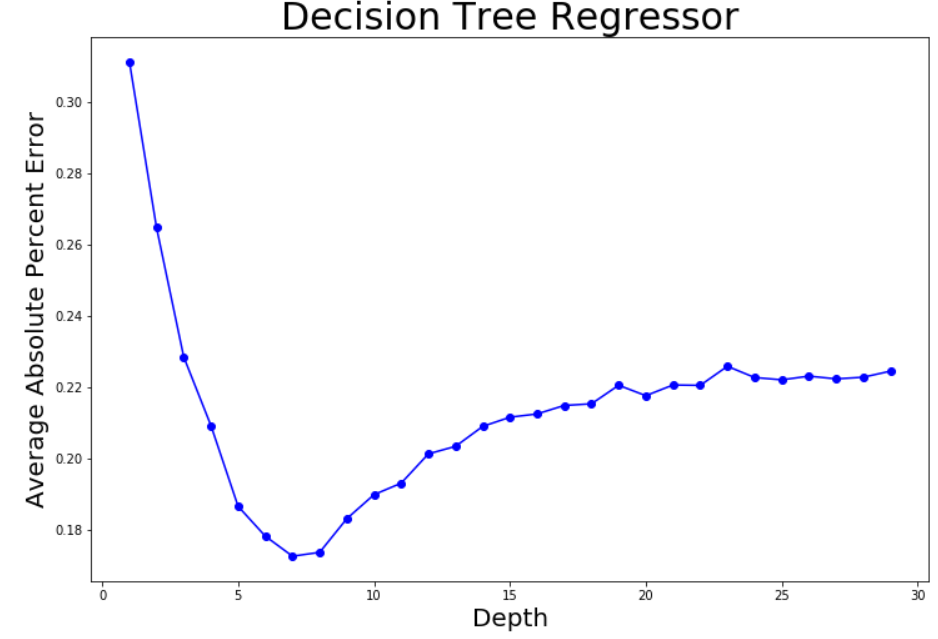
* **Dealing with Missing Values :**

Many ways were tried for dealing with missing values. Those techniques are **Forward Filling, Backward Filling, Mode imputation, Mean Imputation, KNN imputation , MICE imputation**. Out of all these, the most beneficial technique was KNN imputation.

**KNN Imputation:**

Used KNN from fancyimpute library to impute Missing values from nearest matching 3 observations.

* **Machine Learning Models:**
* **Depth=7 was best for all ML Tree based Algorithms.**



After Performing all kinds of ML Algorithms for Regression, I came to conclusion that mostly tree based learning algorithms were performing a lot better than other. An another intuition was all tree based learning algorithms uses weak trees and the optimal depth at which all the algorithms were performing better was at depth = 7.

**What Machine Learning Models were tried and Why I Rejected them :**

**Ordinary Least Squares Regression:** Rejected due to Underfitting

**KNN :** Rejected due to Overfitting

**Support Vector Machine :** Rejected due to Low Performance

**Decision Trees:** Rejected due to Low Performance

**Random Forest:** Used

**Gradient Booster:** Used

**AdaBoost :** Rejected as XGBoost was performing a way better than this.

**XGBoost:** Used

**LightGBM:** Used

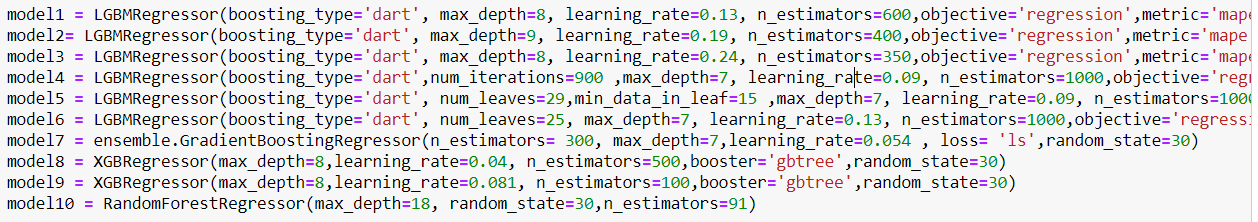
**Perceptron :** Rejected due to Overfitting

**Neural Network :** Rejected due to Overfitting

**What approach was used for reducing Overfitting and make Robust Model:**

**K Fold Cross Validation** technique was used for reducing the model overfitting and dealing with varied data . **5 fold Cross Validation was used**.

**Model Building :**



**Stacking of Models:**

Note that We achieved lowest testing error of 12.9 % from LightGBM model (5th model) but as we had very small amount of data, thus in such cases it may happen that if we use one machine learning model only and it does not deal better with the unseen test dataset, we may get very poor testing accuracy. Thus, We used Stacking approach into account for this type of problem. Thus, we are almost sure that our average of 10 best models will never overfit data. Also, after combining all 10 models prediction, we get testing accuracy of 12.8% which is lower than our best models accuracy and training accuracy reduced to a great extent from 9.06% of our previous best model to just 7.1% which is a lot improvement as we have values in 10e+5 to 10e+6 range.

All 10 models was made from best parameter tuning.

**10 Models which were stacked together are below:**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Model | No | Training Error | 5 Fold Testing Error | No of estimators | Max depth | Learning  rate | Num of iterations | No of leaves | Minimum data in leaf |
| LightGBM | 1 | 0.091082 | 0.130032 | 600 | 8 | 0.13 | 100 | 31 | 20 |
| LightGBM | 2 | 0.091730 | 0.129483 | 400 | 9 | 0.19 | 100 | 31 | 20 |
| LightGBM | 3 | 0.087806 | 0.130125 | 350 | 8 | 0.24 | 100 | 31 | 20 |
| LightGBM | 4 | 0.091184 | 0.130137 | 1000 | 7 | 0.09 | 900 | 31 | 20 |
| LightGBM | 5 | 0.090642 | **0.129125** | 1000 | 7 | 0.09 | 100 | 29 | 15 |
| LightGBM | 6 | 0.086398 | 0.129469 | 1000 | 7 | 0.13 | 100 | 25 | 20 |
| Gradient Booster | 7 | 0.075843 | 0.13391 | 300 | 7 | 0.054 | - | - | - |
| XGBoost | 8 | 0.063819 | 0.1341148 | 500 | 8 | 0.04 | - | - | - |
| XGBoost | 9 | 0.0798823 | 0.1346995 | 100 | 8 | 0.081 | - | - | - |
| Random Forest | 10 | 0.052911 | 0.144339 | 91 | 18 | - | - | - | - |

**Why our final model will never overfit and will be a great Predictor?**

1. We used almost all models which are **Ensemble Learners**, which are the best learners when it comes to learning from weak learners and making a single strong learner. So, even if one learner makes mistake it is not likely that all 1000 learners will make mistake.
2. We used **Stacking** of Different Models. Suppose Random Forest Regressor would not perform better on unseen test data, we have other 9 learners which will not undergo overfitting.
3. We used **Cross Validation** Approach with not large value of K, which may lead to overfitting or thinking that our model is very good and would surely perform better on test data. Thus, instead of much usual value of k=10, I used k=5 which makes sure that we are serving main purpose of K Fold Cross Validation, which is to reduce Overfitting.

Thanks and Kind Regards,

Utsav Patel

[utpatel@iu.edu](mailto:utpatel@iu.edu)

312-774-3172