Project Name

Predictive Analytics on Ames housing data

Project Members:
Mihir Panchal
Omesh Landge
Rajas Sawant
Sumit Satadekar

Statistical language used:

- R programming

Statistical tool used:

- R Studio x64 3.3.3

Project Description:

- Ames Housing Authority is a public housing agency that serves the city of Ames, Iowa, US. It helps provide decent and safe rental housing for eligible low-income families, the elderly, and persons with disabilities

- The housing authority has collected 79 assessment parameters which describes every aspect of residential homes in Ames. These variables focus on the quality and quantity of the physical attributes of a property. Most of the variables are exactly the type of information that a typical home buyer would want to know about a potential property.

- This is a perfect simulation for real-world analytics problem which can tell someone new to housing market as to what variables determines the house sale prices.

Problem Statement:

- Predict Home **Sale Price(dependent variable)** for the Test Data Set with the lowest possible error.

Defining the Raw Datasets:

- There are **4** raw datasets in which **2** datasets has dependent variable **Saleprice** available in it, so it will be my **Train** data and other 2 are **without Saleprice** variable which will be my **Test** Data.

- In **2** datasets of **Train** data, one has **81** variables and another has **76** variables. **81** variables are **without dropping** those columns which has **NA** values with a weightage of more than **40%** and in the dataset with **76** variables these columns with more than **40 % NA** values has been **already dropped**.

- In **2** datasets of **test** data, one has **80** variables and another has 75 variables. **80** variables are without dropping those columns which has **NA** values with a weightage of more than **40%** and in the dataset with **75** variables these columns with more than **40 % NA** values has been already dropped.

- Both the **Train** datasets have all the values of each and every variable **identical**, also in both **Test** datasets all the values of all the variables are **same**. So it is better to keep only **1 Train** and **1 Test** dataset to **avoid the Data Redundancy.**

- We will keep and **work on those 2 datasets which has 81 variables (Train data) and 76 variables (Test data)**, so that we can start the work from scratch which **includes dropping the variables manually with more than 40 % of NA values**.

Data Load and Cleanup:

- Raw Data was imported into **R** environment and was stored in dataframe objects with name **train** and **test**.

- Test data did not have **Saleprice** variable so added the same variable in it. So that we can **merge** both the train and test dataframes into a **single dataframe**.

- Merged both the test and train dataframes into a single dataframe with name **Complete_Data**.

- 5 variables named **Alley, FireplaceQu, PoolQC, Fence, MiscFeature** has more than 40 % of NA values So we will drop these variables completely.

- Total number of rows in each variable of Complete_Data: **2981**

- Following variables of Complete_Data have more than **40 %** of missing values:

Name	**Value**	**Percent**
PoolQC	2909	97.58%
MiscFeature	2814	94.39%
Alley	2721	91.27%
Fence	2348	78.76%
FireplaceQu	1420	47.63%

- Deleted these columns with 40% data missing in "**Complete_Data**" dataframe:
  PoolQC, MiscFeature, Alley, Fence, FireplaceQu

- Complete_Data Summary after deleting above 5 columns:

  Total number of Rows: **2919**
  Total Number of Columns: **76**

- Now, remaining variables which has less than 40 % of NA values will be imputed with the new values by taking either **Mean, Median or Mode**, depending on the **class/type** of variable.

- If the variable has the class of **'factor/categorical'** then we will impute its NA values with the value obtained by taking the Mode of the same variable.

- If the variable has the class of **'numeric/integer',** we first check whether the data in this variable is **Normally Distributed** or **Skewed**. If it is normally distributed then we impute the NA values with the value obtained by taking the **Mean** of the same variable and if it is skewed then we impute the NA values with the value obtained by taking the **Median** of the same variable.

- We check the distribution of data by using multiple R functions like **qqnorm, qqline, describe, Shapiro.test.**

- We made a function called **Mode** which we will need while imputing the values. Also, we made program to find out and impute the NA values directly only to those variables which has class of type **'factor'** in **Complete_Data.**

- Rest, other type of variables like '**numeric/integer'** were imputed manually one-by-one by taking either **mean** or **median** depending on the type of data distribution.

- Few plots/graphs which shows that whether the data of any variable is **Normally distributed** or **Skewed**, which will help to decide how to impute **(mean/median)** the NA values of these **continuous** variables.

```
 plot(Complete_Data$BsmtFinSF2)
qqnorm(Complete_Data$BsmtFinSF2)
qqline(Complete_Data$BsmtFinSF2, col=2)
```

- Here, we can clearly see that my data is **skewed**, as in **X-axis** after the **+1** mark my data starts deviating and changes its paths.

- **'Skew** values less than **1.5** says that the data is **normally distributed** and if it is more than this value than the data is **skewed**.'
- Above, the value is **4.14** which clearly shows that the data is skewed. So, in this case we will impute the NA values by taking the **median** of variable.

```
plot(Complete_Data$TotalBsmtSF)
qqnorm(Complete_Data$TotalBsmtSF)
qqline(Complete_Data$TotalBsmtSF, col=2)
```

- Here, we can clearly see that my data is normally distributed, as in X-axis my data does not deviate much and is very close to the qqline. We can rarely see that some data changes the path and for the time being we can ignore those datapoints and consider the remaining ones, as around **90 %** of data is **normally distributed** and follows the **qqline**.

- Above, the value is **1.16,** as it is less than **1.5** which clearly shows that the data is normally distributed. So, in this case we will impute the NA values by taking the **mean** of variable.

Data Transformation:

- There are only few changes which we are going to perform while transforming the data.

- We will make the changes in variables like **CentralAir** and **MSZoning**.

- The data in **CentralAir** variable is in the form ‘**Y’** and ‘**N’**. So we will change it in a form of **1** and **0**, where **1** means **‘Y’** and **0** means ‘**N’**.

- One type of data in **MSZoning** has value like ‘**All (C)**’, So we changed it to **‘C’** to get rid of the parenthesis ‘**()**’ and to avoid any errors while building a model when this variable is used.

- Some variables generates **error** due to the **empty** values in the dataframe of test and train after they were splitted. So just **imputed** the values in all of them one by one by taking the **Mode** and imputing it at that particular **level**, like:

- **Exporting/Saving** the dataframe into local drive in **CSV** format.

Feature Selection:

- We built some models first using **1** variable. There are few models where we used only **2** variables.

- In the end we used best **14** variables to build the models, So there are different models with the different variables included. The **14** best variable which were used are:

- A new dataframe called **NewFullData** was created using these 14 variables.

- Dataframe NewFullData was split into another sets of train and test data with names **NewTrain** and **NewTest**, with the weightage of **70 %** and **30 %** respectively.

- These new train and test data were used to build the model where we were supposed to use only above mentioned **14** variables.

- There are many places where we have used only **LotArea** and **OverallQual** variables to build models, as these both factors becomes essential while determining the **Saleprice** of any land because **LotArea** decides the price directly but **OverallQual** determines the price indirectly and affects the price of land.

- One always wants to live in a good quality of House. Also, **OverallQual** correlates well with the variables **LotArea** and **Saleprice** with values of **0.093** and **0.55** respectively.

- This makes OverallQual the **important reason** behind the decrease or increa

Algorithms for Statistical Models:

○ Here, first we will store the **Complete_Data** in new dataframe called **NewCompeleteData** and split the **NewCompleteData** into the **test** and **train** dataframes.

○ **70 %** data will be my train data and **30 %** data will be my test data.

1. Decision Tree(DT):

○ In decision analysis, a decision tree can be used to **visually and explicitly represent decisions and decision making**. As the name goes, it uses a **tree-like model of decisions**. Though a commonly used tool in data mining for deriving a strategy to reach a particular goal, it's also widely used in Machine Learning.

○ A decision tree is drawn upside down with its **root** at the top.

○ Here, the Decision tree which was built is **Regression tree, as my dependent variable is 'continuous'.**

○ Decision tree gives few **important variables** after it is built. Above are the **important variables** that were used while making a Regression decision Tree.

○ We also **pruned** the tree just to improve the performance of tree and make it look more elegant when visualised.

○ A tree was pruned at **5** best variables, as at this point the tree gave the least error when checked using **cv.glm** function.

○ se in Saleprice of house along with LotArea.

2. Linear Model(LM):

- First, we built only simple linear model by using **LotArea**.
- Later, we built simple linear model by using **OverallQual**.
- None of them were performing well in terms of **Rsqd** and **MSE**. So tried using different interaction terms like poly, log, etc to check if model performs well.
- Model **MulLM.fit3** is the better model, as it has the highest **Rsqd** of **0.3442** and the lowest **MSE 2.640** compared to the other models.
- Also, after plotting it shows that the response data and fitted data coordinates pretty well and fits nicely in the graph. **Normal Q-Q plot** in it shows that the
- Tried using different **interaction terms** to build different models and concluded that the Model **MulLM.fit3** fits the data nicely.
- Later, I build the model using all the variables, named as **MulLM** and this model performed very well compared to the other linear models.
- Model **MulLM** gives good accuracy of fitting the data with **79.88 %** when used all the variables to build the model.

3. Random Forest(RM):

- Random Decision Forest corrects the decision trees habit of **over fitting** the training data.

- At first only **11** variables were taken to build the RandomForest with name RanFor1 and **500** trees, which are by default taken by the RandomForest.

- Later, number of variables and trees were decreased to **8** and **25** respectively and model was named as **RanFor3** model was built. At the end, variables and trees were reduced to **6** and **25** respectively.

- RandomForest named as **RanFor4** gave the best accuracy and performed well which has **6** variables and **25** trees.

- From these all models, Model **RanFor4** consistently gives the better accuracy of **97%** approximately of fitting the data.

- Here, we can see that as we decrease the number of variables and the size of tree together in a same model, the model gives less error every time.

- Since, the value of error in output changes every time as it selects the different-different types of best **6** variables for each tree while building a RandomForest. But Model RanFor4 gives the least error and hence better accuracy of around 97 % of fitting the data.

- But, the problem here is that it gives the accuracy of **97 %** which is **very big number** and clearly shows that it **over fits** the data. So for this we tried out hands on **SVM** and check its performance.

4. Support Vector Machine (SVM):

- In Machine Learning, SVM are supervised learning models with associated learning algorithms used for **classification** and **regression** analysis.

- SVM are basically used for **Linear Classification** of data, but can also be used for **non-linear** classification using the **kernel trick.** In kernel trick for non-linear classification, kernel implicitly maps the inputs into high dimensional feature spaces.

- Here, first we built the SVM model by using all the **14 best variables** which we selected earlier and named the model as **SVM_mod**.

- This model didn't perform well as it gave more error. So later we went to build the **SVM_mod1** model using only **LotArea** variable. This model performed good as it gave the error of only **5.24**, which is good.

- Variables **LotArea** and **OverallQual** has the strong relation with the dependent variable Saleprice that we checked earlier by using **cor** function and among themselves as well, so we used these variables to build 3rd model called as **SVM_mod2**.

- Model **SVM_mod2** gives the lowest error of **3.43** when build the model using 2 independent variables of LotArea and OverallQual, which shows that this model is good compared to the remaining 2 for fitting the data and there is very less difference between the actual and predicted Saleprice values.

- Here, LotArea and OverallQual both has the significant impact on the Saleprice and shows that both of them should be always preferred when model is built and saleprice is estimated.

Model Selection and Conclusion:

- From these **4** different Model types, we conclude that the model made with **Support Vector Machine** named as **SVM_mod2** is the **best model** compared to all of them, as it gives the less error and good accuracy.

- Also, since we have **more** number of Variables in our dataframe so it is better to use **SVM** model.

- Another reason for choosing this model is that we can **tune** SVM.

- **Tuning** helps as it allows us to specify the **different parameters** in **ranges** for **Gamma** and **Cost**.

- Tuning trials all the combinations of **parameters** and locate the one combination that gives the best **result**.

- Best result is something where the model has the **optimal performance** on data and generates **less error.**

- Hence, due to these all factors and advantage of SVM, we choose **SVM_mod2** model to train and test our data and to estimate the value of Saleprice.

- **Radial Kernel trick** was used as it was more suitable for the data that we have and for the regression analysis of **Saleprice** it performed well as the number of support vectors(around **1200**) it used that were good for my model and its performance.

References:

- https://en.wikipedia.org/wiki/
- https://stackoverflow.com/
- www.analyticsvidya.com/
- https://www.kaggle.com/
- https://www.r-bloggers.com/
- www.youtube.com