House Price Prediction

Kaggle Competition: https://www.kaggle.com/c/house-price-predictioniiitb/leaderboard

Team Members:

Sravanti Nomula (MT2018524)

Sarvesh Nandkar(MT2018519)

PROBLEM STATEMENT

In this project, we will develop and evaluate the performance and the predictive power of a model trained and tested on data collected from houses in Ames, Iowa. Once we get a good fit, we will use this model to predict the monetary value of a house.

ABSTRACT

In the project we have mainly focused on pre-processing as the data had many null entries and Model building. The missing values have been dealt differently depending upon the feature they belong to. We have dropped features like 'PoolQc' which had nearly all the entries missing. We have imputed the missing values using the probability in categorical features like 'MiscFeature'. We have used groupby feature 'Neighbourhood' to impute. We have simply replaced the missing values with 'none' or '0' in other features. We have used several models for the project from basic ones like linear regression to XGBoost. We have also made models by stacking and ensembling few stand alone models gave lesser error than the ensembled ones but as we wanted the model to be more generalized and less overfitting we went for the ensembled model.

INTRODUCTION

The goal of the project is to use the dataset of 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa to do two things: To gain insights into what parameters can affect the price of a house and to predict the final price of a house when certain parameters are known.

DATA EXPLORATION

The train data comprises of 1321 data points with 81 features.

The test data comprises of 139 data points with 80 features.

(This is due to, of course, the fact that the test data do not include the final sale price information!)

```
print(train.shape)
print(test.shape)
```

```
(1321, 81)
(139, 80)
```

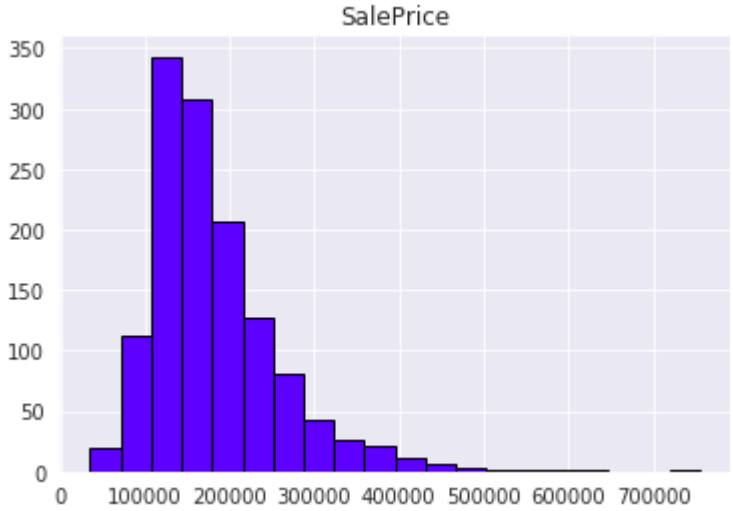
The different features of the data set are:

- SalePrice - the property's sale price in dollars. This is the target variable that you're trying to predict.
- MSSubClass: The building class
- MSZoning: The general zoning classification
- LotFrontage: Linear feet of street connected to property
- LotArea: Lot size in square feet
- Street: Type of road access
- Alley: Type of alley access
- LotShape: General shape of property
- LandContour: Flatness of the property
- Utilities: Type of utilities available
- LotConfig: Lot configuration
- LandSlope: Slope of property
- Neighborhood: Physical locations within Ames city limits
- Condition1: Proximity to main road or railroad
- Condition2: Proximity to main road or railroad (if a second is present)
- BldgType: Type of dwelling
- HouseStyle: Style of dwelling
- OverallQual: Overall material and finish quality
- OverallCond: Overall condition rating
- YearBuilt: Original construction date
- YearRemodAdd: Remodel date
- RoofStyle: Type of roof
- RoofMatl: Roof material
- Exterior1st: Exterior covering on house
- Exterior2nd: Exterior covering on house (if more than one material)
- MasVnrType: Masonry veneer type
- MasVnrArea: Masonry veneer area in square feet
- ExterQual: Exterior material quality
- ExterCond: Present condition of the material on the exterior
- Foundation: Type of foundation
- BsmtQual: Height of the basement
- BsmtCond: General condition of the basement
- BsmtExposure: Walkout or garden level basement walls
- BsmtFinType1: Quality of basement finished area
- BsmtFinSF1: Type 1 finished square feet
- BsmtFinType2: Quality of second finished area (if present)
- BsmtFinSF2: Type 2 finished square feet
- BsmtUnfSF: Unfinished square feet of basement area
- TotalBsmtSF: Total square feet of basement area
- Heating: Type of heating
- HeatingQC: Heating quality and condition
- CentralAir: Central air conditioning
- Electrical: Electrical system
- 1stFlrSF: First Floor square feet
- 2ndFlrSF: Second floor square feet
- LowQualFinSF: Low quality finished square feet (all floors)

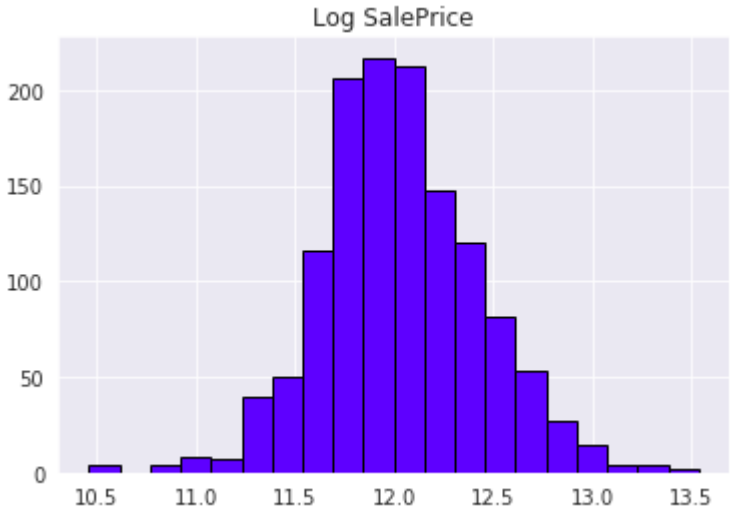
- GrLivArea: Above grade (ground) living area square feet
- BsmtFullBath: Basement full bathrooms
- BsmtHalfBath: Basement half bathrooms
- FullBath: Full bathrooms above grade
- HalfBath: Half baths above grade
- Bedroom: Number of bedrooms above basement level
- Kitchen: Number of kitchens
- KitchenQual: Kitchen quality
- TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)
- Functional: Home functionality rating
- Fireplaces: Number of fireplaces
- FireplaceQu: Fireplace quality
- GarageType: Garage location
- GarageYrBlt: Year garage was built
- GarageFinish: Interior finish of the garage
- GarageCars: Size of garage in car capacity
- GarageArea: Size of garage in square feet
- GarageQual: Garage quality
- GarageCond: Garage condition
- PavedDrive: Paved driveway
- WoodDeckSF: Wood deck area in square feet
- OpenPorchSF: Open porch area in square feet
- EnclosedPorch: Enclosed porch area in square feet
- 3SsnPorch: Three season porch area in square feet
- ScreenPorch: Screen porch area in square feet
- PoolArea: Pool area in square feet
- PoolQC: Pool quality
- Fence: Fence quality
- MiscFeature: Miscellaneous feature not covered in other categories
- MiscVal: $Value of miscellaneous feature
- MoSold: Month Sold
- YrSold: Year Sold
- SaleType: Type of sale
- SaleCondition: Condition of sale

Looking at the data, we see features we expected, like YrSold (the year the home was last sold) and SalePrice. Others we might not have anticipated, such as LandSlope (the slope of the land the home is built upon) and RoofMatl (the materials used to construct the roof). Later, we'll have to make decisions about how we'll approach these and other features.

SALEPRICE



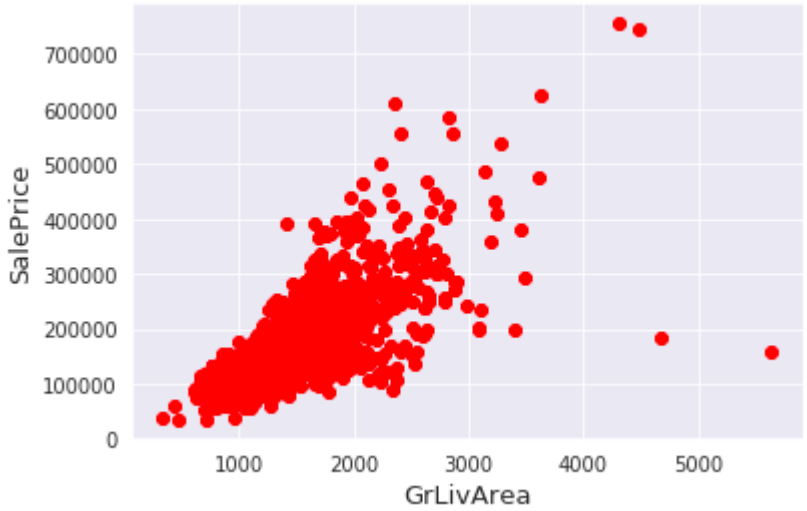
A histogram plot shows the distribution of the target variable ‘SalePrice’ as being was right-skewed. When performing regression, sometimes it makes sense to log-transform the target variable when it is skewed. One reason for this is to improve the linearity of the data.



Importantly, the predictions generated by the final model will also be log-transformed, so we’ll need to convert these predictions back to their original form later.

np.log() will transform the variable, and np.exp() will reverse the transformation.

OUTLIER DETECTION

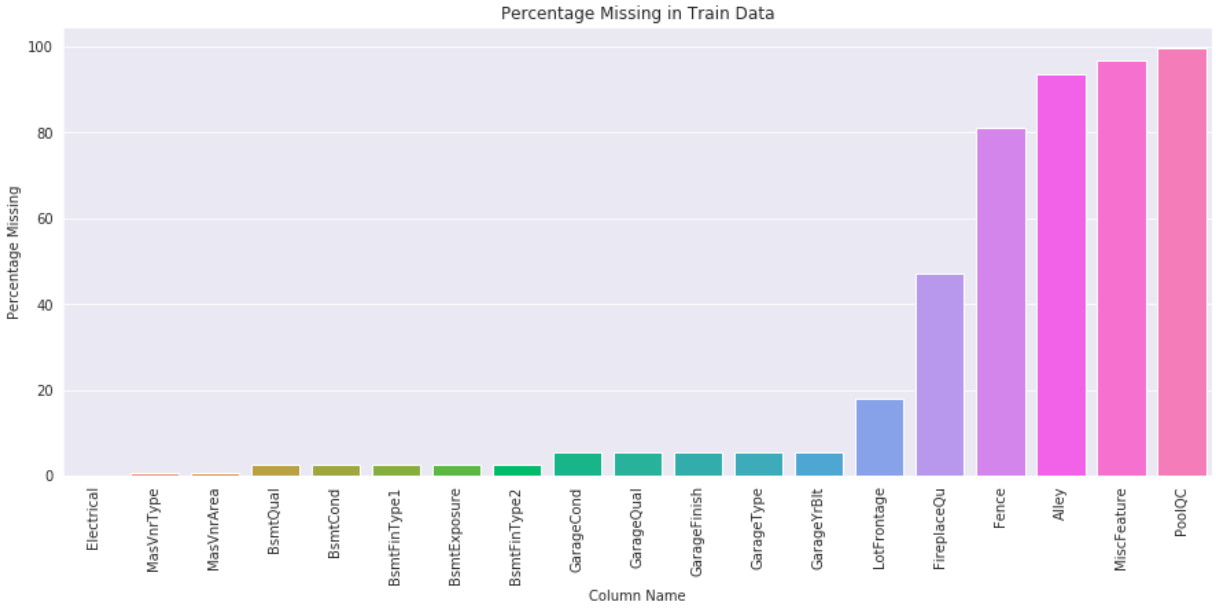


Some of the features had outliers. Looking at the graph above we can see that there are two homes that are very spacious yet are extremely low in price.

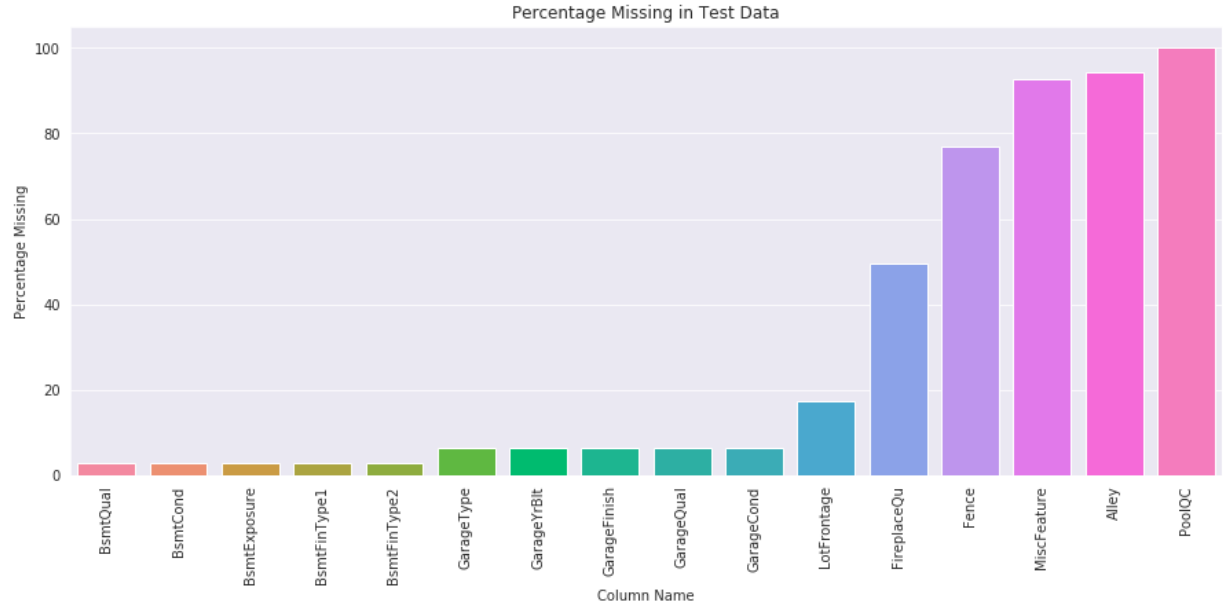
MISSING VALUES

In the train dataset, 19 out of 81 features had missing values. As for the test dataset, 16 out of 80 features had missing values.

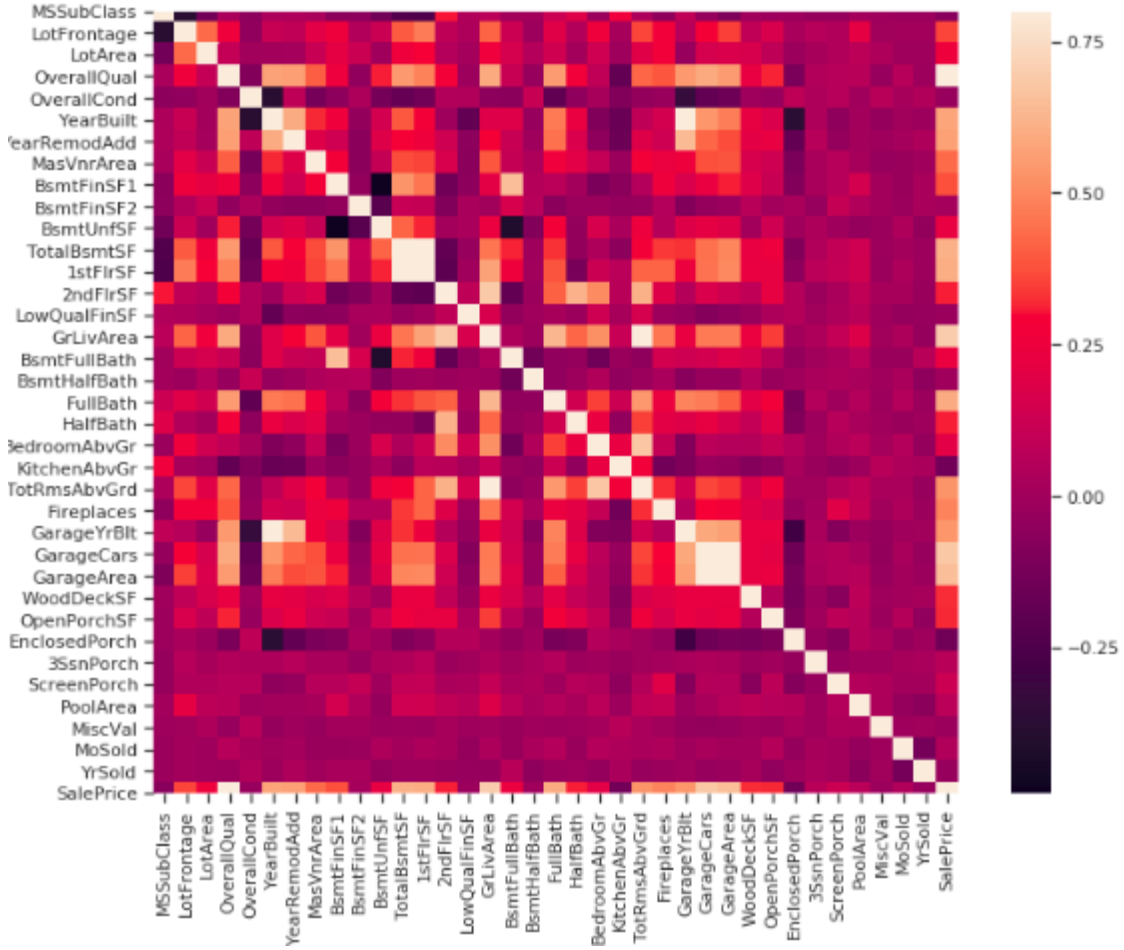
Missing columns in train data,



Missing columns in test data,



CORRELATION MATRIX



Some variables were highly correlated such as 'GarageArea' and 'GarageCars'. This makes sense since the size of a garage determines('GarageArea') the number of cars that fit in it ('GarageCars'). Other variables that were highly correlated show the same type of dependency.

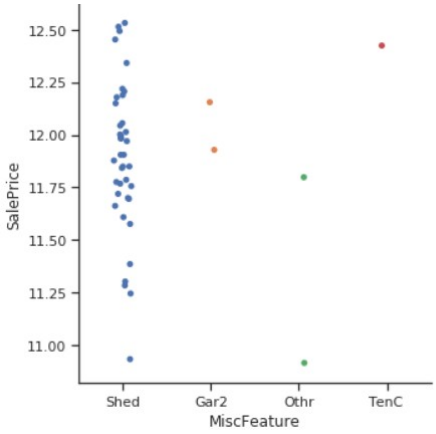
DEALING WITH MISSING VALUES

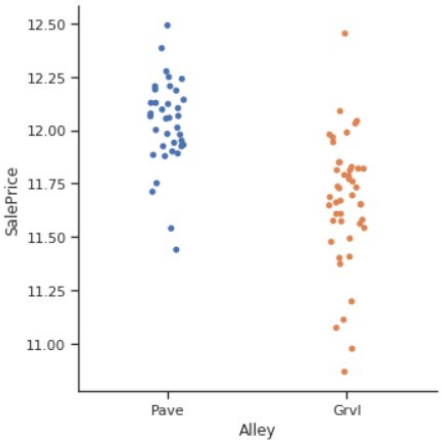
- ***PoolQC:***

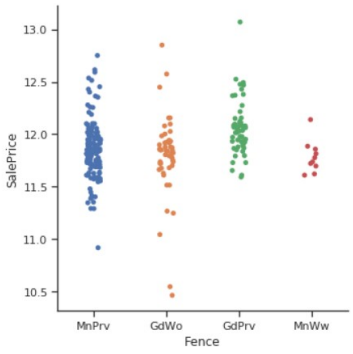
Only 6 non-null entries are available for this feature. So, we have dropped this feature.

- ***Alley, MscFeature, Fence :***

We can see that these categorical features have over 80% of its column missing. However, looking at the data description, the NA's in categorical variables may have actually meant "not present" but at the same time making such an assumption for almost all the data points of these columns may not be a good decision. So, we have imputed all the null entries of these columns with none initially and then again imputed these entries('None') with the help of probability of occurrence of each category of entry.



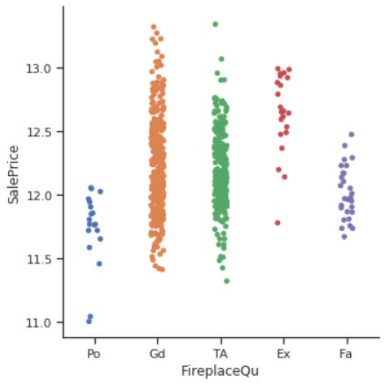




- ***FirePlaceQu:***

In the same way as we imputed the null entries of MiscFeature, Fence, Alley we also imputed the null entries of FirePlace feature.

Firstly, we imputed all the null entries with a categorical entry of ‘None’ then again changed these ‘None’ with the help of probability of occurrence of each category of entry.




```
Before filling :
None    690
Gd      378
TA      312
Fa       33
Ex       23
Po       20
Name: FireplaceQu, dtype: int64

After filling :

Gd      547
TA      459
None    345
Fa       47
Ex       30
Po       28
Name: FireplaceQu, dtype: int64
```

- ***LotFrontage:***

Assuming houses belonging to the same Neighborhood will have similar ‘LotFrontage’, we used groupby Neighborhood and then took mean for each locality. Then we substituted the missing values of a particular Neighborhood with the mean of that Neighborhood.

- ***GarageYrBlt, GarageArea, GarageCar***

As all these features had (<5%) missing values and are numerical we simply imputed ‘0’ in the null entries.

- ***GarageType, GarageFinish, GarageQual, GarageCond***

As all these features had (<5%) missing values and are categorical we simply imputed ‘None’ in the null entries.

- ***Other features***

In the remaining features very, few null entries are present in each entry thus if the feature is categorical and can be ‘None’ we imputed ‘None’ into it and if it is numerical and can be ‘0’ logically we imputed ‘0’ in it.

Some features like Electrical, Exterior1st, Exterior2nd, KitchenQual, SaleType the entries cannot be ‘0’ so we replaced the null entries with mode of non-null entries.

FEATURE ENGINEERING

- Researching further, we realized that few of the numerical features MSSubClass, Mosold, YrSold, MsSubClass, OverallCond were actually categorical. ‘MoSold’ used numerical values represented months. For example, “1” meant January and “2” meant February. In ‘MSSubClass’, numerical values identified the type of dwelling involved in the sale.

  These features were converted into strings.

- Since the data has many categorical variables, we converted them into dummy/indicator variables as linear regression works only on columns with numeric values.
- Checking the correlation of different features with the target variable SalePrice we observe that these features are positively correlated with the SalePrice .

```
: #checking the correlation of features with the target variable
  corrmat["SalePrice"].sort_values(ascending=False)[1:9]

: OverallQual    0.820155
  GrLivArea      0.718499
  GarageCars     0.681676
  GarageArea     0.654091
  TotalBsmtSF    0.645197
  1stFlrSF       0.622147
  FullBath       0.593939
  YearBuilt      0.580423
  Name: SalePrice, dtype: float64
```

- These features are positively correlated with the salePrice hence creating new features by taking 3 polynomials square, cube and square root.

VALIDATION

We have used K-Fold cross validation with k=5. We have used it because it ensures that every observation from the original dataset has the chance of appearing in training and test set. This is one among the best approach if we have a limited input data.

1. Split the entire data randomly into 5 folds (value of k shouldn't be too small or too high, ideally, we choose 5 to 10 depending on the data size). The higher value of K leads to less biased model (but large variance might lead to overfit), whereas the lower value of K is similar to the train-test split approach we saw before.
2. Then fit the model using the 4 (K minus 1) folds and validate the model using the remaining Kth fold. Note down the scores/errors.
3. Repeat this process until every K-fold serve as the test set. Then take the average of your recorded scores. That will be the performance metric for the model.

The scoring function used in K-Fold validation here is MSE which is assumed to be positive usually but comes out to be negative as

- MSE is a loss function
- It is something we want to minimize
- A design decision was made so that the results are made negative.

So, we have negated the score of K-Fold Cross Validation as we want to obtain the RMSE score.

MODEL BUILDING and PREDICTION

Logistic Regression model is used to model the probability of a certain class or event. It uses sigmoid function to build the regression model. Advantage is that it is pretty fast, and the output can be interpreted as a probability. Disadvantage is that it is only fit for linear features and tends to perform poorly when features don't have linear relation with the label.

Random Forests model trains each tree independently, using a random sample of data. This randomness helps to make the model more robust than a single decision tree. It generally doesn't overfits the data in training model, so the results are fairly accurate.

Decision Tree model employs a top-down, greedy search through the space of possible branches with no backtracking which can be used to construct a decision tree for regression by replacing Information Gain with *Standard Deviation Reduction*.

Ridge and *Lasso regressions* are some of the simple techniques to reduce model complexity and prevent over-fitting which may result from simple linear regression.

XGBoost model build trees one at a time, where each new tree helps to correct errors made by previously trained tree. There are typically three parameters - number of trees, depth of trees and learning rate which can be changed as per the requirement. These models are better learners than Random forest models but require significantly more time to train owing to the fact that the trees are built sequentially

LightGBM is a gradient boosting framework that uses tree-based learning algorithms. It is designed to be distributed and efficient with faster training speed and higher efficiency along with lower memory usage and good accuracy.

Lasso, ElasticNet, Kernel Ridge regression, Gradient boost, XGBoost and lightGBM models are used for model building and training the train dataset. Then we applied linear regression, Random forest regression, Bayesian Ridge regression, Decision Tree regressor models individually to the train dataset to find their accuracies.

Averaging models take the mean of individual model predictions and they often reduce overfitting. The basic idea behind stacked generalization is to use a pool of base classifiers, then using another classifier to combine their predictions, with the aim of reducing the generalization error.

After this we applied these models to build an Averaging model using the GBoost, XGBoost and LightGBM models to see the behavior of this combined model. This was more of a learning for us. Then we went on to apply a stacking averaging model using all the regressor models discussed above in hope of getting better a model. And this model predicted quite well with the test part of dataset.

Then applying the boosting models built above individually for prediction we found that these models were much better learners as the theory suggests. So, we combined these models with the stacked averaging models built above and this combined model was then applied to the test_toup dataset provided for the final prediction result. The prediction results are stored in sub.csv.

Below table summarizes the scores we got with different models on training dataset:

Model Name	Score
Lasso	0.1131
Elastic Net	0.1132
Kernel Ridge	0.2747
GBoost	0.1137
XGBoost	0.1185
LGBoost	0.1175
Linear Regression	0.09137
Random Forest	0.05428
Bayesian Ridge	0.10059
Decision Tree	2.575e-05
Averaging model	0.1124

Stacking Averaging model	0.08887
Ensembled model	0.071836

As can be seen from the table, some models give scores of less than the final ensembled score but we haven't considered it alone. This is to reduce any chances of overfitting which can be cases with a single model. Ensembling many models prevents any chances of overfitting.

CONCLUSION

In the project we have mainly focused on pre-processing as the data had many null entries and Model building.

The missing values have been dealt differently depending upon the feature they belong to. We have dropped features like PoolQc which had nearly all the entries missing. We have imputed the missing values using the probability in categorical features like MiscFeature. We have used groupby feature ‘Neighbourhood’ to imputy. We have simply replaced the missing values with ‘none’ or ‘0’ in other features. We have used several models for the project from basic ones like linear regression to XGBoost. We have also made models by stacking and ensembling few stand alone models gave lesser error than the ensembled ones but as we wanted the model to be more generalized and less overfitting we went for the ensembled model.