Under the Guidance of Project by:

Prof. Nik Bear Brown Prasanth Thota

Northeastern University Spoorthi Bellam

Massachusetts, U.S.

**House Price Forecasting using Ensemble and Tensor-flow**

2018

**TABLE of CONTENTS:**

* 1. Abstract 3
  2. Introduction 4
  3. Data set 5

1.4 Methods 6

2.1 Exploratory Data Analysis 6

2.2 Removing Outliers 7

2.3 Target Variable 7

2.4 Missing Values Treatment 8

2.5 Data Correlation 9

3.1 Regression Implementation 10

3.2 Lasso regression 10

3.3 Elastic Net regression 10

3.4 Kernel Ridge Regression 11

3.5 Gradient Boost Regression 11

3.6 XGboost Regression 11

4.1 Ensemble Modelling 12

5.1 Applying Neural Network to tensor flow 15

6.1 Conclusion and Results 17

7.1 Discussion 18

7.2 References 19

**ABSTRACT**

We are working on a project to predict the sales prices of houses. For each Id in the test set,

we should be able to predict the value of the Sales Price variable. We have retrieved data from

Beeviva and did EDA, used various regression techniques, applied ensemble Model and also applied Neural Network for tensor flow to get the best results. To predict the sales prices we have to take RMSE on logs of the predictions and targets, which makes it the RMSLE and have to get that score which is shown below.

People looking to buy a new home tend to be more conservative with their budgets and market strategies. The existing system involves calculation of house prices without the necessary prediction about future market trends and price increase. The goal of the paper is to predict the efficient house pricing for real estate customers with respect to their budgets and priorities. By analysing previous market trends and price ranges, and also upcoming developments future prices will be predicted. The functioning of this paper involves a website which accepts customer’s specifications and then combines the application of multiple linear regression algorithm of Regression and ensemble. This application will help customers to invest in an estate without approaching an agent. It also decreases the risk involved in the transaction.

With 79 explanatory variables describing (almost) every aspect of residential homes in Ames, Iowa, this dataset helps to create a model that will predict the final price of each home. The original dataset was taken from beeviva competition and after performing comprehensive analysis, we enhanced it by adding new predictors that is independent variables related to house features keeping in mind geographical region and existing residents requirements for their prospective house choices.

Since this is a regression problem, we deployed a number of methodologies for getting accurate sales price.

Procedures to be followed:

1. **Understand the problem**. We'll look at each variable and do a philosophical analysis about their meaning and importance for this problem.
2. **Univariable study**. We'll just focus on the dependent variable ('SalePrice') and try to know a little bit more about it.
3. **Multivariate study**. We'll try to understand how the dependent variable and independent variables relate.
4. **Basic cleaning**. We'll clean the dataset and handle the missing data, outliers and categorical variables.
5. **Test assumptions**. We'll check if our data meets the assumptions required by most multivariate techniques.

****

**INTRODUCTION**

This paper brings together the latest research on prediction markets to further their utilization by economic forecasters. Thus, there is a need to predict the efficient house pricing for real estate customers with respect to their budgets and priorities. This paper efficiently analyses previous market trends and price ranges, to predict future prices. This topic brings together the latest research on prediction markets to further their utilization by economic forecasters. It provides a description of prediction markets, and also the current markets which are useful in understanding the market which helps in making useful predictions. Thus, there is a need to predict the efficient house pricing for real estate customers with respect to their budgets and priorities. This paper uses linear regression algorithm to predict prices by analysing current house prices, thereby forecasting the future prices according to the user’s requirements.

The main goal of the model used in the paper is to different values for the dependent variable based on several independent variables by using different regression techniques and then comparing the results and their accuracy by RMSE value and plotting the best results.

We have used 2 of the CSV files namely, train\_users, test\_users as our dataset. Based on the current residents requirement in that region we have added few other attributes and analysed their correlation with the selling price . This is done to meet customers requirement for the features and services in the houses.

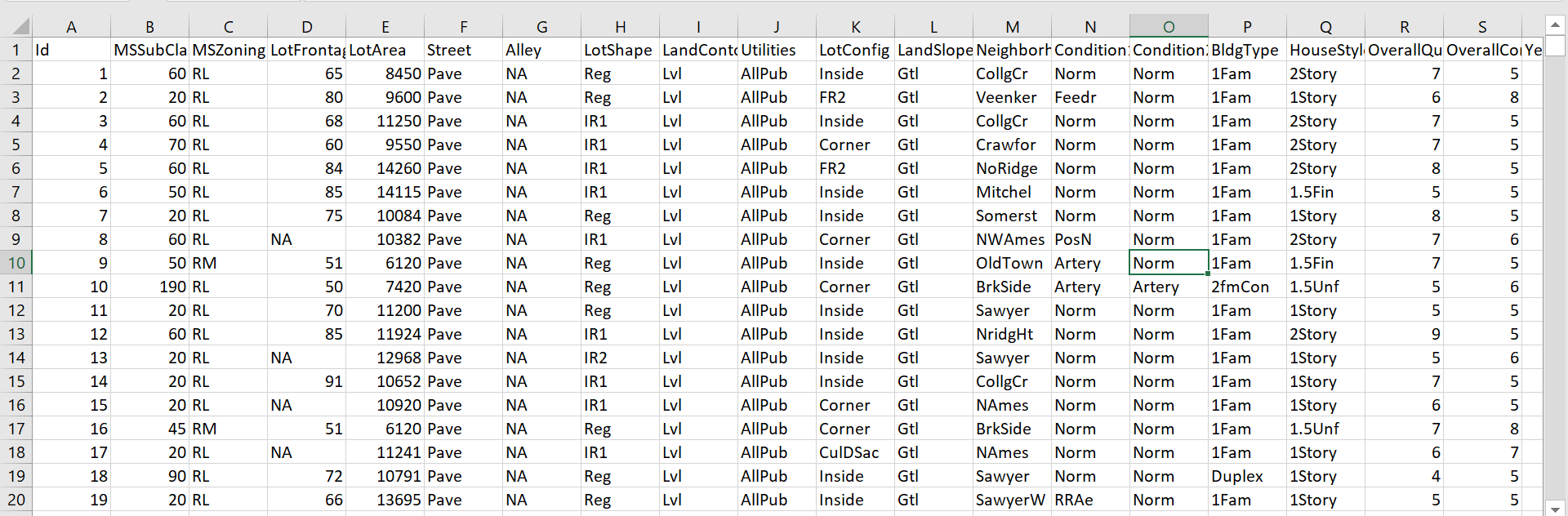
As the dataset contained a number of inconsistent and null values which could have hampered the prediction, we performed data cleaning and dropped the irrelevant entries. Further, Exploratory Data Analysis(EDA) was performed, resulting into statistical insights and visualizations on the trained dataset. For model selection, we started with supervised learning algorithms and then performed ensembling for getting best model out of all.

**DATA SET**

The dataset is downloaded from <http://www.bee-viva.com/competitions/ames>

Numerical Features: 1stFlrSF, 2ndFlrSF, 3SsnPorch, BedroomAbvGr, BsmtFinSF1, BsmtFinSF2, BsmtFullBath, BsmtHalfBath, BsmtUnfSF, EnclosedPorch, Fireplaces, FullBath, GarageArea, GarageCars, GarageYrBlt, GrLivArea, HalfBath, KitchenAbvGr, LotArea, LotFrontage, LowQualFinSF, MSSubClass, MasVnrArea, MiscVal, MoSold, OpenPorchSF, OverallCond, OverallQual, PoolArea, ScreenPorch, TotRmsAbvGrd, TotalBsmtSF, WoodDeckSF, YearBuilt, YearRemodAdd, YrSold

Categorical Features: Alley, BldgType, BsmtCond, BsmtExposure, BsmtFinType1, BsmtFinType2, BsmtQual, CentralAir, Condition1, Condition2, Electrical, ExterCond, ExterQual, Exterior1st, Exterior2nd, Fence, FireplaceQu, Foundation, Functional, GarageCond, GarageFinish, GarageQual, GarageType, Heating, HeatingQC, HouseStyle, KitchenQual, LandContour, LandSlope, LotConfig, LotShape, MSZoning, MasVnrType, MiscFeature, Neighborhood, PavedDrive, PoolQC, RoofMatl, RoofStyle, SaleCondition, SaleType, Street, Utilities

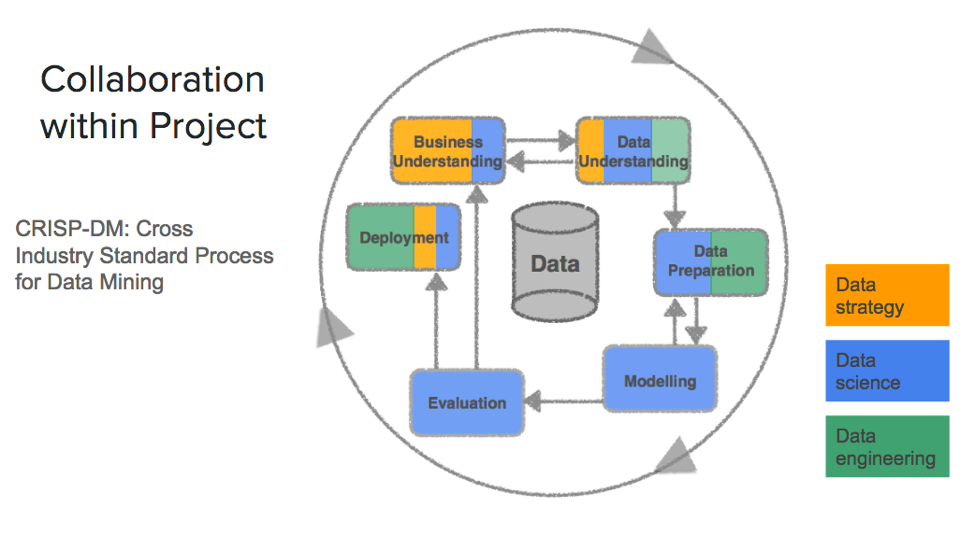


**METHODS**

**Methods/ Algorithms used**:

* Exploratory Data Analysis
* Linear regression model implementation.
* Ensemble.
* Applying neural network using Tensor-flow.

**Exploratory Data Analysis**

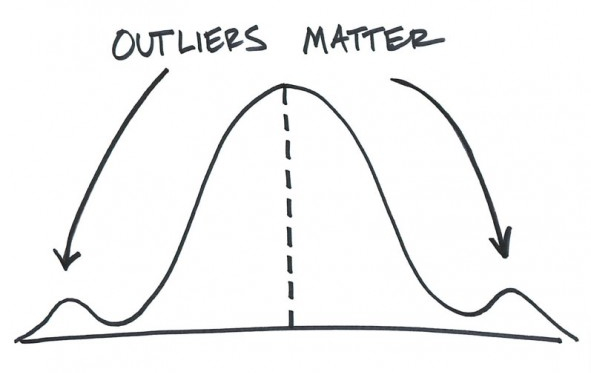


The first step always is to look and explore the available data to understand how it is distributed and what is the relation with other attributes.  In this phase our main aim is to have a better understanding of the features involved in our data. It might be possible that some are left behind but I will be focusing on the features that have the highest correlation towards Sales Price.

1. Removing outliers:

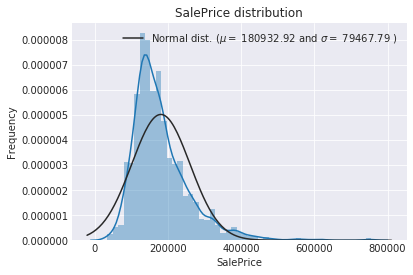


We can see at the bottom right two with extremely large GrLivArea that are of a low price. These values are huge oultliers. Therefore, we can safely delete them.



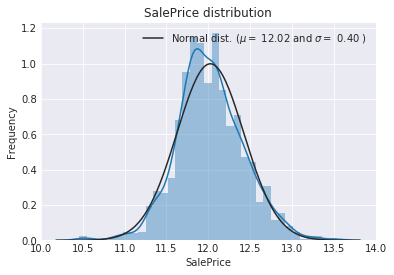
## Target Variable

**SalePrice** is the variable we need to predict. So let's do some analysis on this variable first.

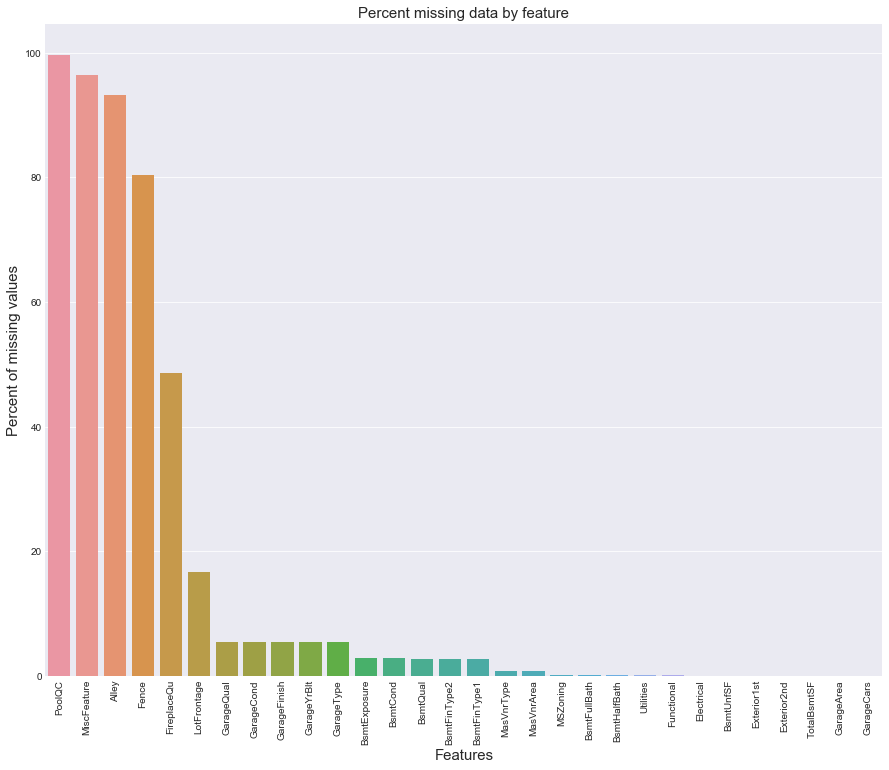


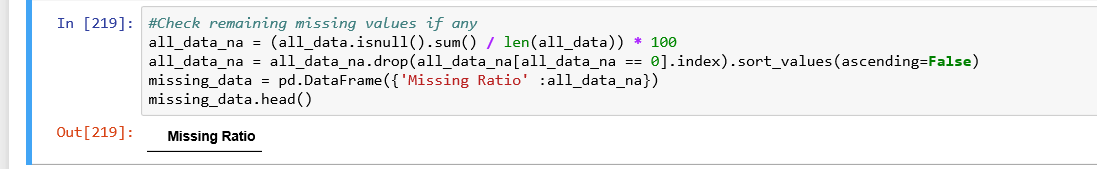
The target variable is right skewed. As (linear) models love normally distributed data , we need to transform this variable and make it more normally distributed.

**Log-transformation of the target variable**

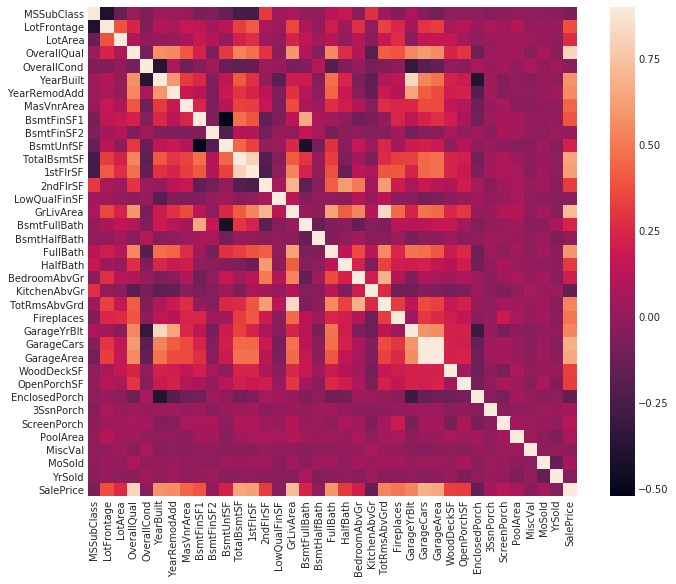


1. Removing the missing values





**Data Correlation:**



The heatmap is the best way to get a quick overview of correlated features thanks to seaborn!

At initial glance it is observed that there are two red colored squares that get my attention.

1. The first one refers to the 'TotalBsmtSF' and '1stFlrSF' variables.
2. Second one refers to the 'GarageX' variables. Both cases show how significant the correlation is between these variables. Actually, this correlation is so strong that it can indicate a situation of multicollinearity. If we think about these variables, we can conclude that they give almost the same information so multicollinearity really occurs.

Heatmaps are great to detect this kind of multicollinearity situations and in problems related to feature selection like this project, it comes as an excellent exploratory tool.

'SalePrice' correlations: As it is observed that 'GrLivArea', 'TotalBsmtSF', and 'OverallQual' saying a big 'Hello !' to SalePrice, however we cannot exclude the fact that rest of the features have some level of correlation to the SalePrice.

**Regression implementation**

Here we have used 5 regression models.

1. Lasso regressoion
2. Elastic Net regression
3. Kernel Ridge Regression
4. Gradient Boost Regression
5. XGboost Regression

Lasso Regression:

This model may be very sensitive to outliers. So we need to made it more robust on them. For that we use the sklearn's **Robustscaler()** method on pipeline

Lasso regression analysis is a shrinkage and variable selection method for linear regression models. The goal of lasso regression is to obtain the subset of predictors that minimizes prediction error for a quantitative response variable. The lasso does this by imposing a constraint on the model parameters that causes regression coefficients for some variables to shrink toward zero. Variables with a regression coefficient equal to zero after the shrinkage process are excluded from the model. Variables with non-zero regression coefficients variables are most strongly associated with the response variable.

lasso = make\_pipeline(RobustScaler(), Lasso(alpha =0.0005, random\_state=1))

Elastic Net regression:

ElasticNet will tend to select more variables hence lead to larger models (also more expensive to train) but also be more accurate in general. In particular Lasso is very sensitive to correlation between features and might select randomly one out of 2 very correlated informative features while ElasticNet will be more likely to select both which should lead to a more stable model (in terms of generalization ability so new samples).

ENet = make\_pipeline(RobustScaler(), ElasticNet(alpha=0.0005, l1\_ratio=.9, random\_state=3))

Kernel Ridge Regression:

Kernel ridge regression (KRR)  combines [Ridge Regression](http://scikit-learn.org/stable/modules/linear_model.html#ridge-regression) (linear least squares with l2-norm regularization) with the kernel trick. It thus learns a linear function in the space induced by the respective kernel and the data. For non-linear kernels, this corresponds to a non-linear function in the original space.

KRR = KernelRidge(alpha=0.6, kernel='polynomial', degree=2, coef0=2.5)

Gradient Boost Regression:

**Gradient boosting** is a [machine learning](https://en.wikipedia.org/wiki/Machine_learning) technique for [regression](https://en.wikipedia.org/wiki/Regression_(machine_learning)) and [classification](https://en.wikipedia.org/wiki/Classification_(machine_learning)) problems, which produces a prediction model in the form of an [ensemble](https://en.wikipedia.org/wiki/Ensemble_learning) of weak prediction models, typically [decision trees](https://en.wikipedia.org/wiki/Decision_tree_learning). It builds the model in a stage-wise fashion like other [boosting](https://en.wikipedia.org/wiki/Boosting_(meta-algorithm)) methods do, and it generalizes them by allowing optimization of an arbitrary [differentiable](https://en.wikipedia.org/wiki/Differentiable_function) [loss function](https://en.wikipedia.org/wiki/Loss_function).

With **huber** loss that makes it robust to outliers

GBoost = GradientBoostingRegressor(n\_estimators=3000, learning\_rate=0.05,

max\_depth=4, max\_features='sqrt',

min\_samples\_leaf=15, min\_samples\_split=10,

loss='huber', random\_state =5)

XGboost Regression:

The library is laser focused on computational speed and model performance, as such there are few frills. Nevertheless, it does offer a number of advanced features.

model\_xgb = xgb.XGBRegressor(colsample\_bytree=0.4603, gamma=0.0468,

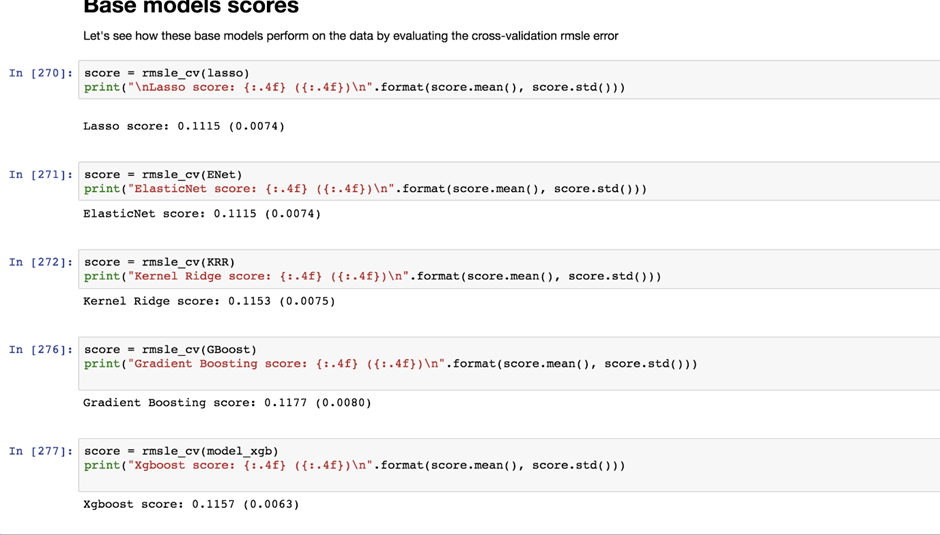
learning\_rate=0.05, max\_depth=3,

min\_child\_weight=1.7817, n\_estimators=2200,

reg\_alpha=0.4640, reg\_lambda=0.8571,

subsample=0.5213, silent=1,

random\_state =7, nthread = -1)

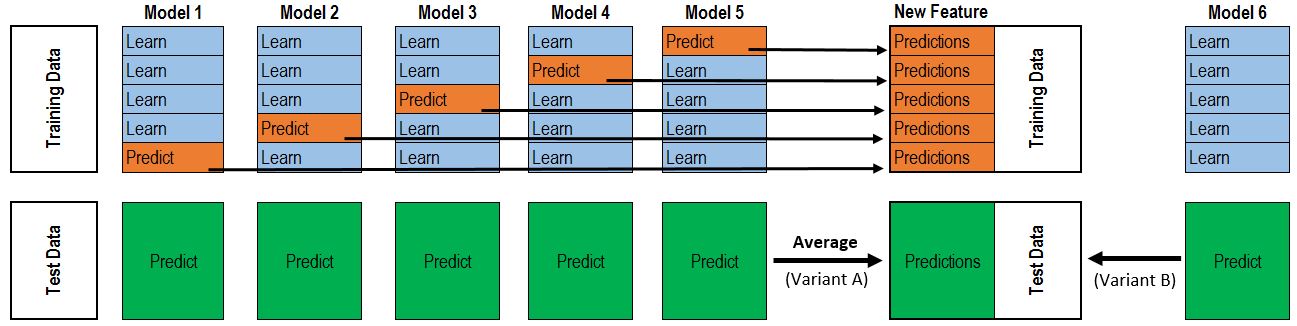


The above are the scores for the models that we used. We have used root mean square error for evaluation.

**Root mean square error**: The root-mean-square deviation (RMSD) or root-mean-square error (**RMSE**) (or sometimes root-mean-squared error) is a frequently used measure of the differences between values (sample or population values) predicted by a model or an estimator and the values observed.

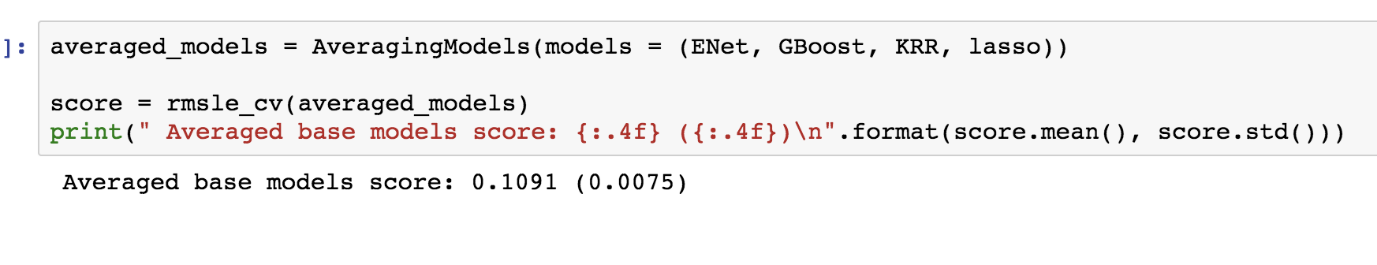
**Ensemble Modeling**

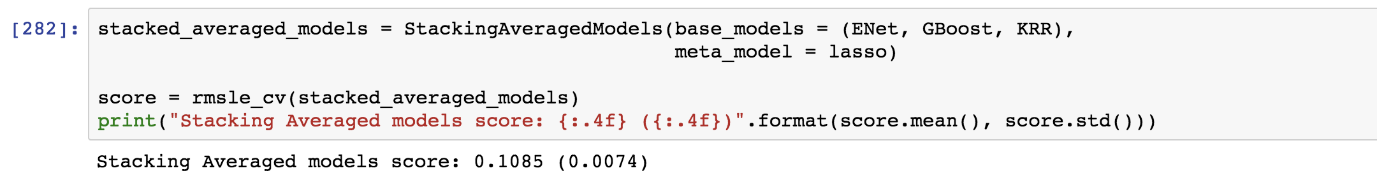
Ensemble modeling is a powerful way to improve the performance of your model. It usually pays off to apply ensemble learning over and above various models you might be building. Time and again, people have used ensemble models in competitions like Kaggle and benefited from it. Ensemble learning helps improve machine learning results by combining several models. Ensemblemethods are meta-algorithms that combine several machine learning techniques into one predictive model in order to decrease variance (bagging), bias (boosting), or improve predictions (stacking)



We have two types of ensembles. (i). Average stacking, (ii) Stacking Ensemble.

Less simple Stacking : Adding a Meta-modelIn this approach, we add a meta-model on averaged base models and use the out-of-folds predictions of these base models to train our meta-model. The procedure, for the training part, may be described as follows: Split the total training set into two disjoint sets (here train and .holdout )Train several base models on the first part (train)Test these base models on the second part (holdout)Use the predictions from 3) (called out-of-folds predictions) as the inputs, and the correct responses (target variable) as the outputs to train a higher level learner called meta-model. The first three steps are done iteratively. If we take for example a 5-fold stacking , we first split the training data into 5 folds. Then we will do 5 iterations. In each iteration, we train every base model on 4 folds and predict on the remaining fold (holdout fold). So, we will be sure, after 5 iterations , that the entire data is used to get out-of-folds predictions that we will then use as new feature to train our meta-model in the step 4.For the prediction part , We average the predictions of all base models on the test data and used them as meta-features on which, the final prediction is done with the meta-model.





we see that the results have improved after the second stacking.

**Applying Neural Network to tensor flow**

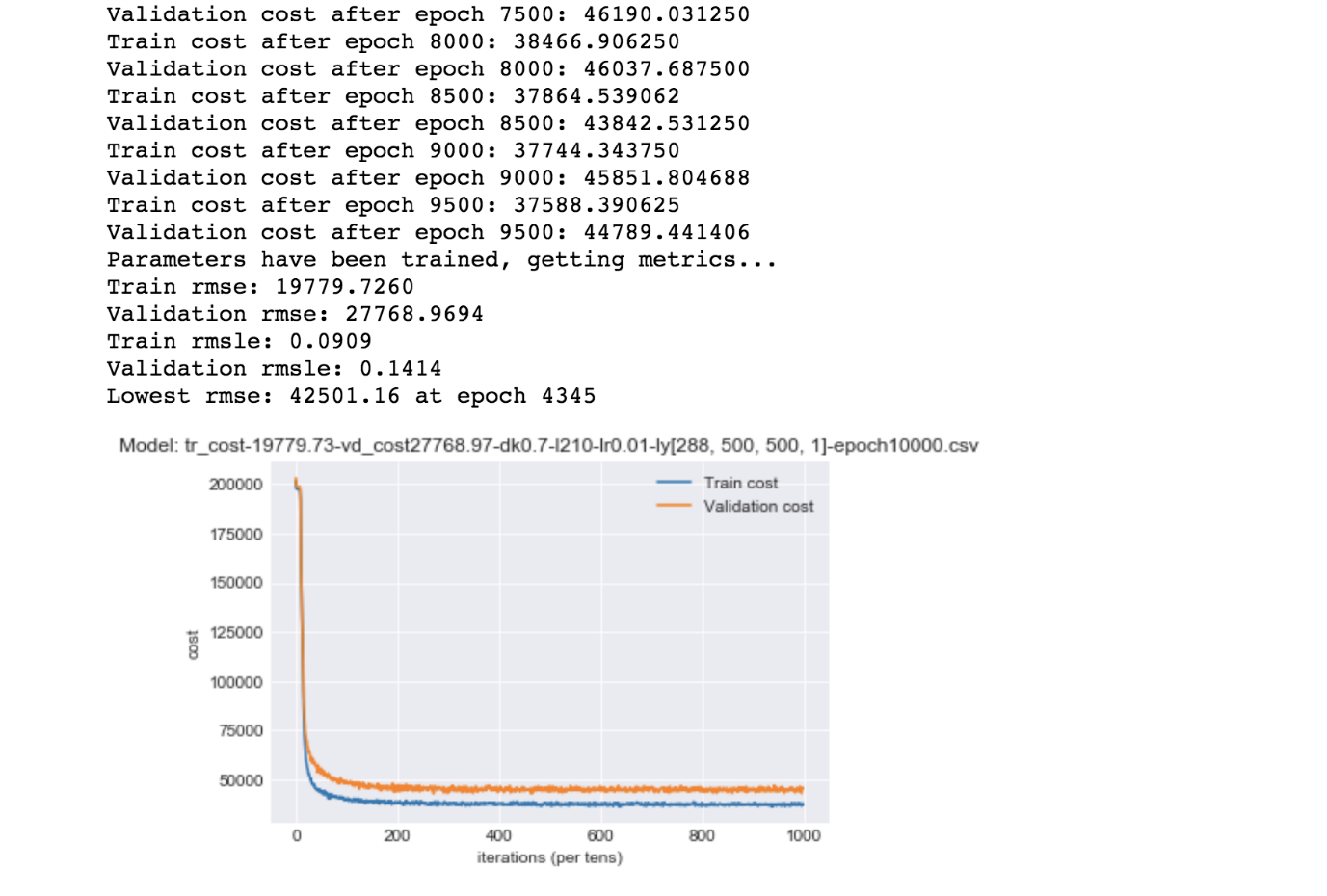
TensorFlow is an open source software library for numerical computation using data flow graphs. Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them. The flexible architecture allows you to deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API.

Tensorflow is a computational framework for building machine learning models. TensorFlow provides a variety of different toolkits that allow you to construct models at your preferred level of abstraction. You can use lower-level APIs to build models by defining a series of mathematical operations. Alternatively, you can use higher-level APIs (like tf.estimator) to specify predefined architectures, such as linear regressors or neural networks.



A machine learning application is the result of the repeated computation of complex mathematical expressions. In TensorFlow, a computation is described using the Data Flow Graph, where each node in the graph represents the instance of a mathematical operation (multiply, add, divide, and so on), and each edge is a multi-dimensional data set (tensors) on which the operations are performed.

TensorFlow uses a **dataflow graph** to represent your computation in terms of the dependencies between individual operations. This leads to a low-level programming model in which you first define the dataflow graph, then create a TensorFlow **session** to run parts of the graph across a set of local and remote devices.



We printed the least rmse for the number of epoch i.e 4335. Then we save the csv of the predictions.

**CONSULSION and RESULTS**

## The description mentions to get the sales prices RMSE on logs of the predictions and targets, which makes it the RMSLE values so we found the RMSLE value to get the best scores.

## We see that the stacking which has a Rmsle error =0.0074 has greater performance compared to average ensemble which has Rmsle error= 0.0075 Because stacking takes a calculated weightage whereas averaging takes an average of all.

* The averaging method is not intuitive and thus it’s not efficient.
* First simplest stacking approach really improved the score. This encouraged us to go further and explore a less simple stacking approach.
* We philosophised about the variables, we analysed 'Sales Price' alone and with the most correlated variables, we dealt with missing data and outliers, we tested some of the fundamental statistical assumptions and we even transformed categorial variables into dummy variables. That's a lot of work that Python helped us make easier.
* I wanted to focus mainly on **feature engineering** and the **stacking** technique. I think stacking is a very useful tool to have within your Data Science toolkit.
* Evaluation for base model scores:

- Lasso score: 0.1115 (0.0074)

- ElasticNet score: 0.1115 (0.0074)

- Kernel Ridge score: 0.1153 (0.0075)

- Gradient Boosting score: 0.1177 (0.0080)

- Xgboost score: 0.1157 (0.0063)

- LGBM score: 0.1162 (0.0071)

Of all the models we see that Xgboost has the least error. That means its the best model.



**DISCUSSION**

In today’s real estate world, it has become tough to store such huge data and extract them for one’s own requirement. Also, the extracted data should be useful. The system makes optimal use of the Linear Regression Algorithm. The system makes use of such data in the most efficient way. The linear regression algorithm helps to fulfil customers by increasing the accuracy of estate choice and reducing the risk of investing in an estate. A lots of features that could be added to make the system more widely acceptable. One of the major future scopes is adding estate database of more cities which will provide the user to explore more estates and reach an accurate decision. More factors like recession that affect the house prices shall be added. In-depth details of every property will be added to provide ample details of a desired estate. This will help the system to run on a larger level.

From EDA , we were able to find missing values , outliers , skews. And further replacing text data with none and numerical with 0 or median depending upon attribute. Rectification of skew to normalized plot was done. The house features that are most helpful and important for this prediction are Fire place, garage area, hall area etc. These attributes are highly correlated with our dependent variable that is selling price.Thus we removed the lease correlated attributes and also avoided redundancy of the features in the model.

**References**

[1] Vishal Raman, May 2014. Identifying Customer Interest inReal Estate Using Data Mining.

[2] <http://www.99acres.com/property-rates-and-pricetrendsin-mumbai>

[3] Douglas C. Montgomery, Elizabeth A. Peck, G. Geoffrey Vining, 2015. Introduction to Linear Regression Analysis

[4] Gongzhu Hu, Jinping Wang, and Wenying FengMultivariate Regression Modeling for Home ValueEstimates with Evaluation using Maximum Information Coefficient

[5] Iain Pardoe, 2008, Modeling Home Prices Using Realtor Data

[6] Aaron Ng, 2015, Machine Learning for a Lond

on Housing Price Prediction Mobile Application

[7] <https://www.tensorflow.org/guide/graphs>



THANK YOU

