Machine Learning Project Report

Title: House Prices: Advanced Regression Techniques

Team:

1. Vinay Potla – vxp170930
2. Sruthi Kotaru – sxk178631
3. Divya Teja Pamisetty – dxp172730

Introduction

House prices prediction and valuation is one of the most important trading decisions. A home buyer will not mention the height of the basement ceiling or the proximity to an east-west railroad as the requirements for his house. In this project, we used a dataset which proves that much more influences the price negotiation that the number of bedrooms or a white-picket fence. The dataset contains 79 explanatory variables with 1460 training instances and 1458 test instances describing almost every aspect of residential homes in Ames, Iowa. We are using this dataset to predict the final price of each home. This is a regression problem. The training data is preprocessed and algorithms such as the multilevel perceptron regressor, random forest, perceptron, and linear regression are used to predict the house prices when a set features are given. The error metric used is Root Mean Square Error and the data visualization is done using Seaborn Heatmaps and Matplotlib. All the algorithms are implemented in python. We used many python libraries like scikit learn to develop algorithms, pandas and numpy for data preprocessing, and data frames, one of the data structures of the pandas package to read and manipulate data.

Problem Definition

Task Definition

In this project, we are trying to predict the price of a house using different machine learning algorithms when requirements of a house to be bought are given. Technically, we use the training data on different algorithms and use the test data to find the accuracy and finally when a new instance is given, we predict the value. This is a regression problem.

Algorithm Definition

I. Random Forest Regression and Grid Search:

- This model is more advanced, as it uses multiple trees (decision trees) and will give us the prediction mean of each tree. The model is kind of like a black box, as this mechanism is vague because it gives out the results but does not give any information of the weights.

- The model does not require assumptions such as normality, it (algorithm) is robust and not prone to outliers.

- In this method, each feature node is split by the one which is the best among the set of predictors at a node. By doing so, this method turns out to work well compared to other classifiers.

- This model is a very easy where it depends only on two parameters which are number of variables and number of trees. The algorithm works as follows.

  Step 1: Takes n bootstrap samples from the train data

  Step 2: For each of these samples, make a tree which is unpruned with the modification, at a node instead of splitting the best among all predictors, we sample the predictors randomly and will choose the best among those.

  Step 3: The new data is predicted by taking the prediction of n trees by majority of the votes.

II. AdaBoost Regression:

- AdaBoost, which is short for adaptive boosting, where the outputs of other weak learning algorithms are combined as the weighted sum which produces the final output of this classifier. This approach is adaptive, as the weak classifiers are tuned for the instances which are misclassified.

- This method is very sensitive to noise and outliers. But, as we are combining the weak learners and pro- duce the prediction of the data makes it to a strong learner. Unlike other classifiers such as Neural Networks and Support Vector Machines, this algorithm considers only those features which helps in the increase the model prediction, also reduces the dimensionality and execution time as we consider only features which improves the power of model as the features which are irrelevant is not computed.

- Algorithm:

Given the set of examples $\{(x1, y1), \ldots, (x_m, y_m)\}$ where x_i belongs to domain and y is class label which is -1, +1

Step 1: Initialize a distribution D = 1/m

Step 2: Iterate over t where t = 1 to T,

- Train each weak classifier h using the distribution D
- We get the hypothesis with error
- Choose alpha
- Update weights

The final hypothesis of the classifier will be sign function of summation of each weak classifier multiplied with alpha. The shortcomings of this model are identified by the high weight data points.

III. Gradient Boosting Regression:

- Gradient boosting model, is a technique which gives out the prediction of the model by combining all the weak prediction models. It is done by using a stage-wise manner and will optimize the loss function.

- Gradient boosting model, will train the model sequentially and in stage-wise manner. For each new model, it reduces the loss function of the model by using GD (gradient decent) method. This boosting method is combination of gradient decent and boosting algorithms.

- Algorithm:

  Step 1: We consider mean as the prediction of the features.
  Step 2: Finding out the error of each latest prediction.
  Step 3: Consider variables, which will split the misclassified points perfectly. **Step 4**: After splitting, find the errors of each on both sides (on the split variable).

  Step 5: Keep repeating step 3 and 4 until we reach the objective function

  Step 6: Find out the mean with weights of all classifiers to produce the final output of the model.

IV. K-Nearest Neighbors Regression:

- K-NN is a lazy algorithm and non-parametric.

- We say it is non-parametric, as don't need to consider any assumptions on the data distribution and lazy algorithm as unlike algorithms we don't generalization by using training data. Hence the training of the model is very fast. In the training phase, it just stores the examples.

- In the test phase, take a vote by considering its k-nearest neighbors for its classification.

- Algorithm:

  Step 1: Consider the train data

  Step 2: We train the model by storing the instances by the learner

  Step 3: We find out the k-similar instances to the test data point from the train dataset

  Step 4: The predicted class label is the majority value of the labels.

Experimental Evaluation:

Methodology:

Dataset and Features:

The data set contains 1460 train data instances and 1458 test data instances with 79 attributes describing the different aspects of a home in Ames, Iowa. It contains two files, train.csv and test.csv. There is no separate validation set. The training data has a final value for the house price under the attribute 'Sale Price'. The model should be trained on this training data. The data has been preprocessed before training the model.

Correlation:

(a) Preprocessing of the data:

- The null values in the data are preprocessed.
- The text data has been replaced with numbers (0, 1, 2, …) to convert them to categorical data.
- The numerical data has been standardized, i.e., subtracting the mean from each of the values and dividing by the standard deviation.

(b) Model Building:

i. Training and building the model:

- The preprocessed training data can be used to train the Artificial Neural Networks model. The weights for the neural network are initialized randomly and number of hidden layers are manually checked and a considerably working model based on regression is built in a few iterations.

- For better training, we used Random Forest algorithm, one of the best ensemble methods. The data is split for training and validation using 10-fold stratified cross validation technique.
- We found that the model can be made better by choosing a set of desired features instead of all 79. The feature engineering is done by eliminating unnecessary features.

ii. Training data:

- The training dataset has been divided into training set and validation set. 80% of the data is used as training data and the remaining 20% for validation. Different models such as Artificial Neural Networks, Linear Regression and Random Forest have been trained using the training data.
- The model has been trained using the top 29 most correlated features, in other words, top 29 features having the highest correlation values with the 'Sale Price' feature are used for training.

iii. Feature Engineering:

- The problem maps to the real-world scenario of estimating the price of a house, given different parameters briefing the condition of it. The goal of this project is to build a model that can predict the price of a house. Below are the various steps that were performed:
    - **Understanding the data:** This step results in better feature engineering. The data is understood in its philosophical and analytical point of view. Some of the initial studies include:

        a. Knowing importance of each variable
        b. Analyzing the distribution of dependent and independent variables in the data

        In our case the dependent variable is 'Sale Price' and the rest are independent variables.

    - **Refining the data:** The data was found to have a great deal of missing values. Besides, many independent features/variables were found to be redundant (conveying same information). In addition to this more than half of the variables were either categorical or of type string. Refining data eradicates all the anomalies mentioned above and brigs down data to a minimalistic set of features that carry most of the information required for the prediction. The steps for cleaning the data are as follows:

- **Removing missing values:** There are three ways to reduce/remove missing values. They are:
  - Deleting the data rows that have missing values
  - Deleting the columns that have missing values more than a threshold
  - Interpolation

A mix of above three methods has been used to clean the data. It was found that variables <'PoolQC', 'MiscFeature', 'Alley', 'Fence', 'FireplaceQu', 'LotFrontage','GarageCond', 'GarageType', 'GarageYrBlt', 'GarageFinish','BsmtExposure', 'BsmtFinType2', 'BsmtFinType1', 'BsmtCond','MasVnrArea', 'MasVnrType'> had a lot of missing throughout the dataset rows.

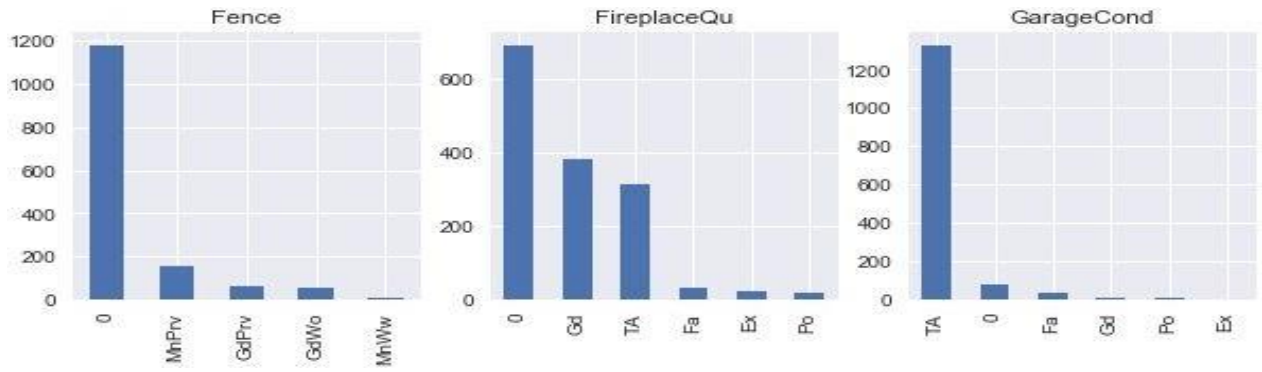
These variables were extracted as per the criteria that purges the variables from the data having more than 15% of the values missing. Besides, for the variable Electrical, only one row had a missing value and row was deleted since one data-point does not make much of a difference.

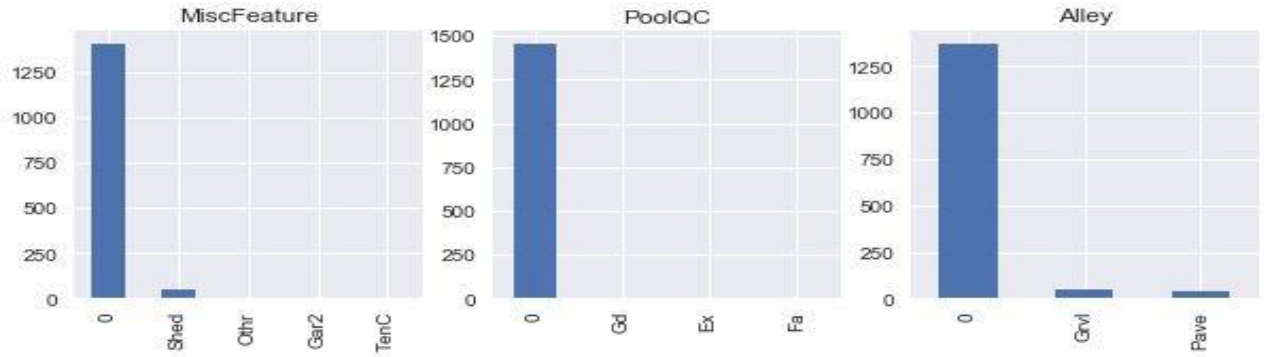
For the rest of the missing values, if the variable is of type int, then the missing value is replaced with the mean of the column values else it is replaced with the mode of column values.

- **Removing redundant columns**: First, the correlation values of all the independent variables with the dependent variable 'Sale Price' are plotted plotted

  GarageCars and GarageArea conveyed same info and both are strongly correlated to SalePrice. Removed GarageArea as its correlation with SalePrice was lesser than that of GarageCars.
  TotalBsmtSF and 1stFloor are almost the same things. Removed 1stFloor.

- **Choosing best variables**: The top (= 29, 20, 15) variables that showed higher correlation with 'Sale Price' that others, were chosen for training the model. The model showed better accuracy when the feature count was 29.





The above graphs show that the data is not uniformly distributed and most of these features have null values. On this basis, these features can be eliminated.

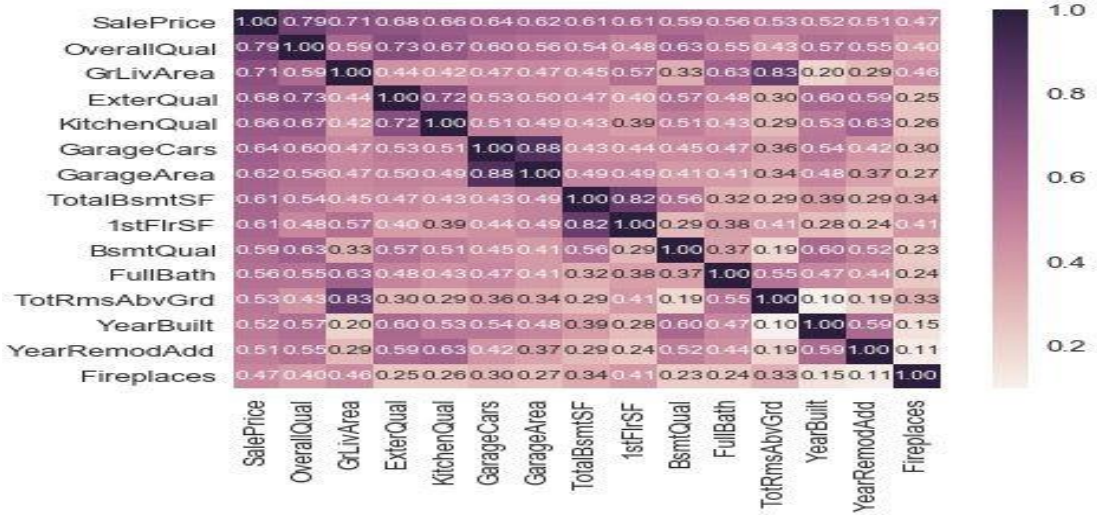
iv. **Validation Data**:

As mentioned, 20% of the data has been used for validation and tested with various algorithms.

(i) Correlation between the feature and Sale price have been calculated. The top 15 features having the most correlation value have been selected as the features:

Features: ['SalePrice', 'OverallQual', 'GrLivArea', 'ExterQual', 'KitchenQual', 'GarageCars', 'GarageArea', 'TotalBsmtSF', '1stFlrSF', 'BsmtQual', 'FullBath', 'TotRmsAbvGrd', 'YearBuilt', 'YearRemodAdd', 'Fireplaces', 'HeatingQC']

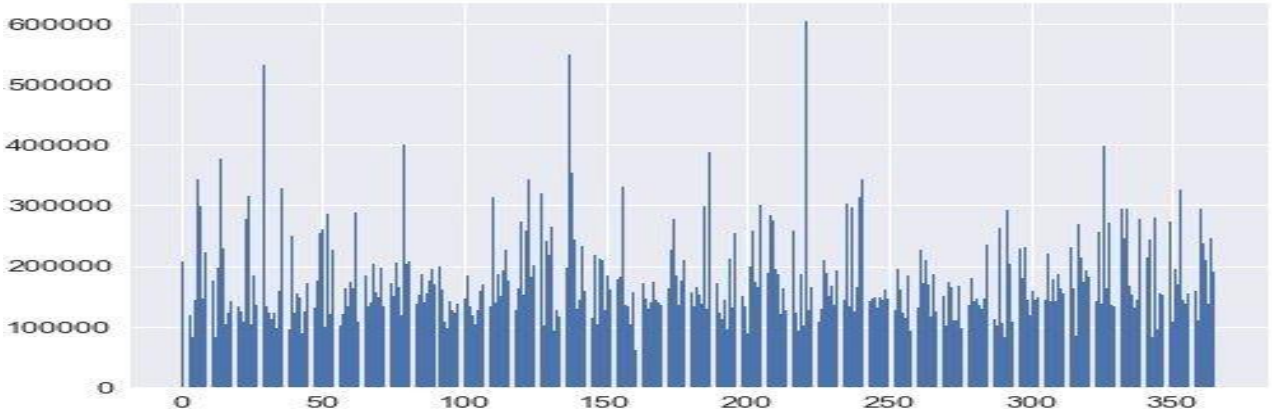
(ii) Top 30 best correlated features:



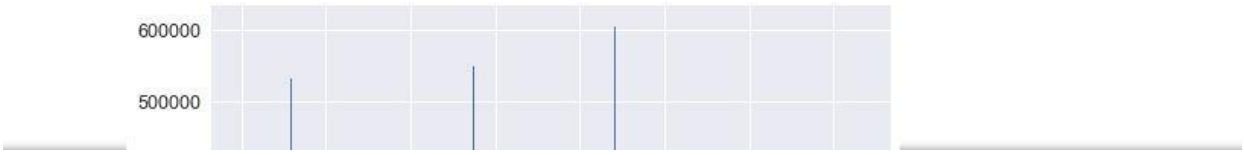
Classifiers:

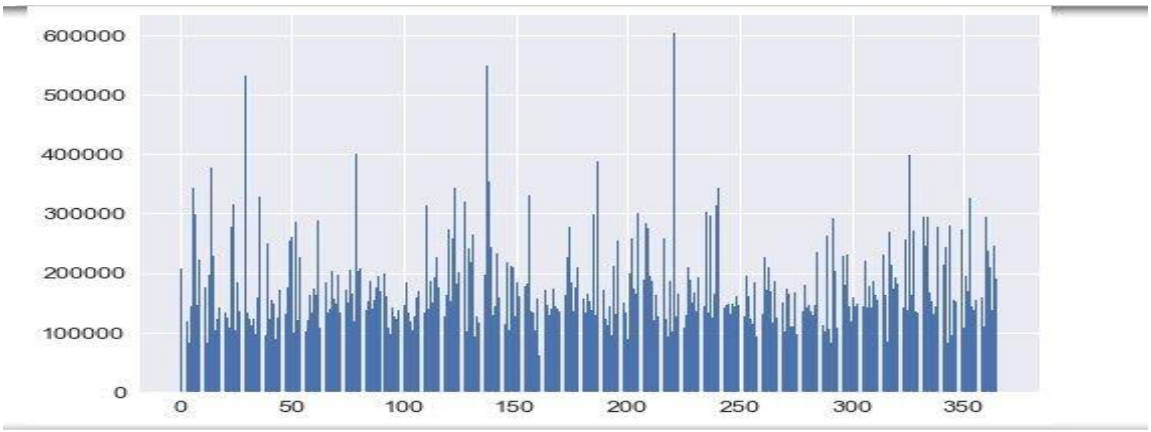
Random Forest:

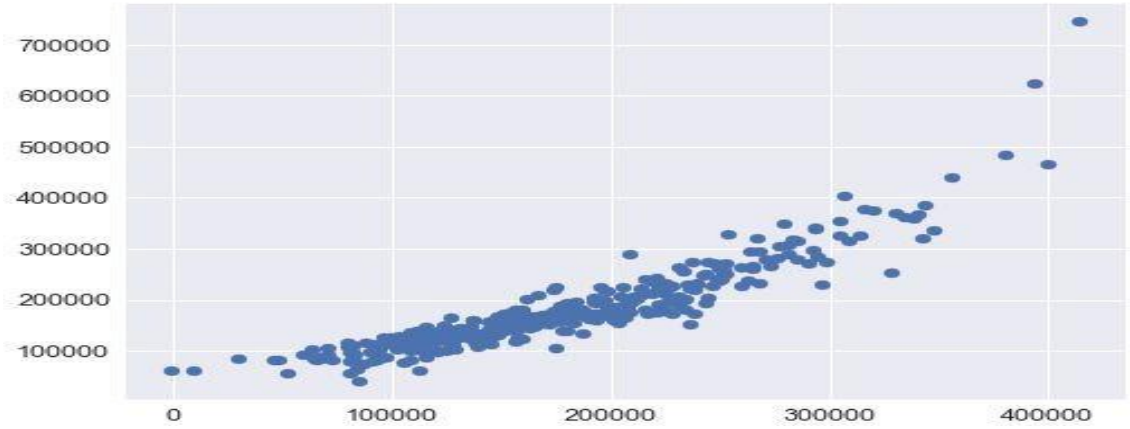
Score: 0.8878472405544457
RMS: 0.16042106469803125




```
  141311.4  186721.6  163660.   154390.   104645.   231002.8  162100.
   85150.   269548.   213260.   174555.   192790.   183083.5  166450.
  142130.   257190.3  136810.   397735.2  162242.1  271700.3  134860.
  134175.   127880.   295600.   245071.   295080.2  166657.5  153091.4
  130380.   143200.   277084.7  167015.5  214680.   242850.    82930.
  279442.4   94730.   154273.7  152400.   140910.   273734.9  108780.
  195250.   169238.5  327255.   144195.   137050.   154930.   188973.5
  158595.   109550.   295515.2  238105.6  210082.8  138130.   246782.2
  190270. ]
Score : 0.9102591625923739
RMS : 0.14986641443063167
ut[100]: <Container object of 365 artists>
```



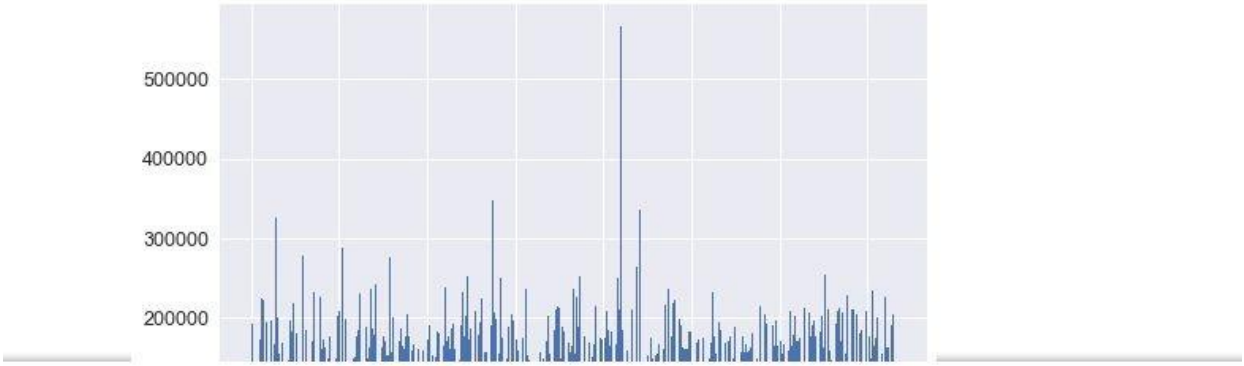




Other Classifiers:

```
Score : 0.4934854332437926
RMS : 0.2800685648476791
Model : MLPRegressor(hidden_layer_sizes=(30))
```

```
ut[111]: <Container object of 365 artists>
```



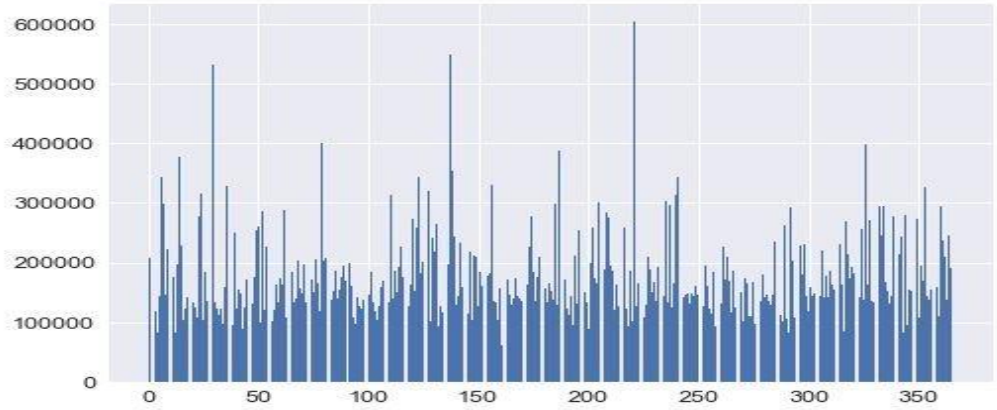
```
Score : 0.6163320005608495
RMS : 0.26486899858316054
Model : MLPRegressor(hidden_layer_sizes=(100))
```

```
ıt[110]: <Container object of 365 artists>
```


```
Score : 0.6332510974465345
RMS : 0.2587172143551888
Model: MLPRegressor(hidden_layer_sizes=(110))
```

```
[109]: <Container object of 365 artists>
```

```
Score : 0.8313423423823133
RMS : 0.14986641443063167
Model: Linear Regression
]: <Container object of 365 artists>
```



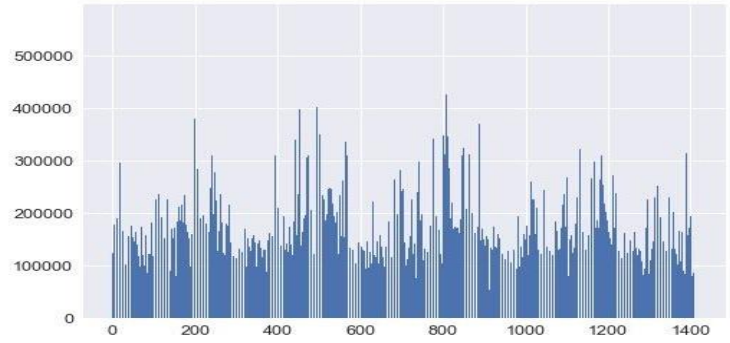
v. Testing data:

The model must give one value for the 'Sale Price', given a new instance based on the test data. The performance measure can be evaluated using mean square error.

Test data Set:

The dataset test.csv, is loaded. The null values are removed, and the categorical values are change to numerical values. We found that the results were obtained by Random Forest algorithm. Hence, we used the same classifier on the test data as well.

```
In [94]:
1 test_predictions = rfr.predict(test_data)
2 test_predictions
3 plt.bar(range(1,test_predictions.shape[0]+1),test_predictions)
Out[94]: <Container object of 1413 artists>
```



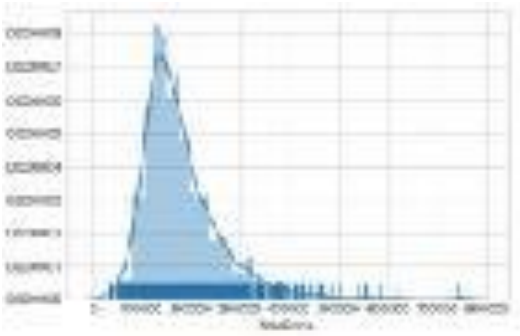

```
In [95]:
1 test_predictions
Out[95]: array([ 123640. ,  160465. ,  185290. , ...,  169045.9,  129010. ,
        234767.8])
```

V. Principle Component Analysis (Dimensionality Reduction):

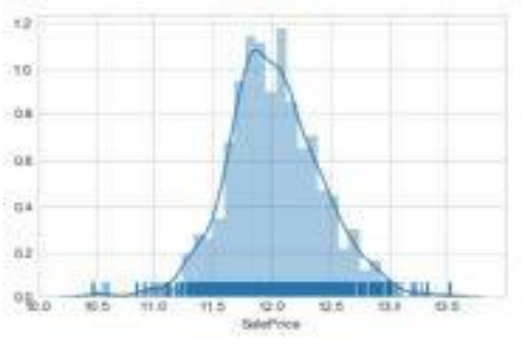
- Principle Component Analysis is used reduce the number of dimensions (attributes) from the dataset as not all collected features are independent.
- Many attributes are correlated, so by combining these features we can reduce the dimensionality of the data set while keeping the number of data points same which helps in better prediction without redundant information.
- To get these new features we try to form new features which are orthogonal to each other and are in the direction of maximum variance in the data.
- We used PCA to reduce number of features to 90, 70, etc., from around 300 features and adopted the one which gave best desirable results.

Log transformation of target variable:

Sale Price before log transformation:



Sale Price after log transformation:



Results:

a. Correlation:

The following results have been obtained by running the regression algorithms on the validation dataset which is every time determined randomly from the training data through 10-fold cross validation. 29 features have been used to tabulate the results.

Classifier	Best Parameters used	Accuracy	RMS error
Random Forest	n_estimators=10 n_jobs=1	91.03	0.149866414430
Linear Regression	copy_X=True fit_intercept=True n_jobs=1 normalize = False	83.04	0.148004201789
ANN (Hidden layer size = 100,50)	activation='relu' alpha=0.0001 Hidden_layer_size = 100, 50	75.92	0.211260796379
ANN (Hidden layer size = 15,30,10)	activation='relu' alpha=0.0001 Hidden_layer_size = 15, 30, 10	72.96	0.223932696117
ANN (Hidden layer size = 110)	Activation = 'relu' alpha=0.0001 Hidden_layer_size = 110	63.51	0.254165151362
ANN (Hidden layer size = 100)	activation='relu' alpha=0.0001 Hidden_layer_size = 100	61.93	0.263212275876
ANN (Hidden layer size = 80)	activation='relu' alpha=0.0001 Hidden_layer_size = 80	58.16	0.268590832304
ANN (Hidden layer size = 60)	activation='relu' alpha=0.0001 Hidden_layer_size = 60	56.40	0.280703734187
ANN (Hidden layer size = 50)	activation='relu' alpha=0.0001 Hidden_layer_size = 50	54.41	0.277564406074
MLP Regressor	activation='relu' alpha=0.0001 hidden_layer_sizes=30	48.66	0.288040023594

b. <u>Principle Component Analysis</u>:

Random Forest Regressor:

Trained the Random Forest model on the training set with hyperparameters. GridSearchCV is used to find the best parameters.

Best Parameters:

Base Model: Decision Tree

Criterion: 'mse'

Maximum Depth of each iterator: 300

Evaluation Metric: Root Mean Squared Error (RMSE)

Result: RMSE between log transformed target variable (predicted) and the actual value is 0.1801

Gradient Boosting Regressor:

Trained Gradient Boosting Regressor on training set with various hyperparameters. GridSearchCV is used to find the best parameters.

Best Parameters:

Alpha: 0.9

Criterion: 'friedman_mse'

Learning Rate: 0.1

Total number of estimators: 300

Evaluation Metric: Root Mean Squared Error (RMSE)

Result: RMSE between log transformed target variable (predicted) and the actual value is 0.151.

AdaBoost Regressor:

Trained AdaBoost regressor on training set with various hyper parameters. Used GridSearchCV to find the best parameters.

Best parameters:

Loss: 'linear'

Learning rate: 0.1

Total number of estimators: 300

Evaluation metric: Root Mean Squared Error (RMSE)

Result**:** RMSE between log transformed target variable(predicted) and the actual value is 0.21

K - Nearest Neighbor Regressor:

Trained K Nearest Neighbor regressor on training set with various hyper parameters. Used GridsearchCV to find the best parameters.

Best parameters:

Metric: 'minkowski'

Number of neighbors: 10

Evaluation metric: Root Mean Squared Error (RMSE)

Result: RMSE between log transformed target variable(predicted) and the actual value is 0.24

Future Work

As mentioned above, preprocessing is the most fundamental step for this problem. For this, we need to do more feature engineering which might give us more information in predicting the sales price. Due to time limitations, the models can also be tuned with some more different parameters which gives us the best accuracy for prediction. We can also use more advanced models and techniques like stacking different Ensemble models etc. The main goal of this project is to understand the regression techniques and feature engineering. By considering this project, we would like to try out these mod- els with more complex machine learning problems and evaluate their performance.

Conclusion

We tried out different models and found out that the best model which fits the data is Gradient boosting which gave RMSE value 0.151(87% accuracy) and concluded by evaluating all the different models using root-mean squared error as the evaluation metric.

Bibliography

[1] https://www.coursera.org/learn/ml-foundations/lecture/2HrHv/learning-a-simple-regression-model-to-predict-house-prices-from-house-size

[2] http://www.sciencedirect.com/science/article/pii/S0957417414007325

[3] https://yalantis.com/blog/predictive-algorithm-for-house-price/

[4] Housing Value Forecasting Based on Machine Learning Methods, Jingyi Mu, Fang Wu, and Aihua Zhang

[5] https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data

[6] http://scikit-learn.org/stable/tutorial/basic/tutorial.html