House Price Prediction Using Machine Learning Techniques

Rishabh Mehta
801257231

ITCS5156 Fall 2022

12th October 2022

Abstract

This report is presented as a survey of previous work [4][5]. Any assertions made within are subjective and do not represent those of the original author.

This paper studies the different machine learning techniques to predict house prices. The data set that is used is provided by Dean De Cock [1], which contains the sale of individual residential properties in Ames, Iowa. The following papers are used for this study:

De Cock, D. (2011). Ames, Iowa: Alternative to the Boston housing data as an end-of-semester regression project. Journal of Statistics Education, 19

Kangane, Pranav & Mallya, Aadesh (2021). Analysis of Different Regression Models for Real Estate Price Prediction. International Journal of Engineering Applied Sciences and Technology

Contents

1 Introduction

Purchasing a house is one of the most important financial decisions in a person's life. People are very cautious about it. With the ever-increasing data on the sale of houses, machine learning can be applied to predict the house price with precision. Price prediction will enable people to make informed decisions.

1.1 Problem statement

- Goal of the project is to build an end-to-end solution or application that can predict house prices better than individuals.
- Purpose of this project is to deepen the knowledge of regression methods in machine learning.

1.2 Motivation

- Unaffordability of housing has become one of the major challenges for metropolitan cities around the world.
- When deciding the sale price of the house, usually people look at comparable properties in the area to decide the sales price, which may not be a correct indicator of the actual price.
- To gain a better understanding of the commercialized housing market we are currently facing, we want to figure out what are the top influential factors of the housing price.

1.3 Approach

A machine learning model is proposed to predict house prices based on data related to the house. This project compares the different machine learning models XGBoost, Support vector regression, Random Forest, Decision tree, etc., and compares their performance using Mean Absolute Error. The dataset is obtained from Kaggle [5] website. To clean the data, I used data preprocessing and preparation techniques, built machine learning models able to predict house prices based on house features, and analyzed and compared models' performance to choose the best models.

2. Related Works

2.1 House Price Prediction Using Multilevel Model and Neural Networks - Feng and Jones (2015)

This paper illustrates the use of a multi-level model (MLM) and Artificial Neural network model (ANN) to model house prices and compared them to each other and the hedonic price model (HPM). The dataset used in this paper is Greater Bristol between 2001 and 2013.

MLM combines the micro-level equation, describing the within-neighborhood between-house relationship for individual properties, and the macro-level equation, the between-neighborhood relationship, into one model.
To enable model training and testing, the dataset was divided into a training set that contains data about house sales from 2001 to 2012, and a test set that contains data about house sales.

The three models were tested using three scenarios. In the first scenario, location and measured attributes were not included in the data. Grid references for house locations were provided in the data in the second scenario. Measured neighborhood characteristics were included in the data in the third scenario. The models were compared in terms of goodness of fit (R), mean absolute error (MAE) and mean absolute percentage error (MAPE), and explanatory power. MLwiN software was used to fit HPM and MLM models, while IBM SPSS software was used to fit ANN models. MLM model outperforms other models in general.

2.2 House Price Prediction: Hedonic Price Model Vs Artificial Neural Network

The objective of this study was to compare the predictive power of a hedonic model with an artificial neural network model on house prices. The study was done on a sample of 200 houses in Christchurch, New Zealand.

According to the hedonic pricing theory, a commodity like a home may be thought of as an aggregation of individual components or qualities. Hedonic price theory originates from the proposal that goods are inputs in the activity of consumption, with a product of a set of characteristics.

The artificial Neural Network model consists of three main layers: input data layer (for example the property attributes), hidden layer(s) (commonly referred to as "black box"), and output layer (estimated house price). A neural network is an interconnected network of artificial neurons with a rule to adjust the strength or weight of the connections between the units in response to externally supplied data

The hedonic price model involves regressing attributes hypothesized to contribute to the price of a house to observed asking prices of the house. The Artificial Neural network is like the process utilized in building the hedonic price model however the neural network is first trained from a set of data. For a particular input, an output (estimated house price) is produced from the model. Then, the model compares the model output to the actual output

The results of this study conclude that the artificial neural network can overcome some of the problems related to data patterns and the underlying assumption of the hedonic model. R2 of neural network models is higher than the R2 of hedonic price models, and the RMSE of neural network models is lower than hedonic price models. Therefore, it can be concluded that the neural network model is a relatively superior model for house price prediction

2.3 In Relation

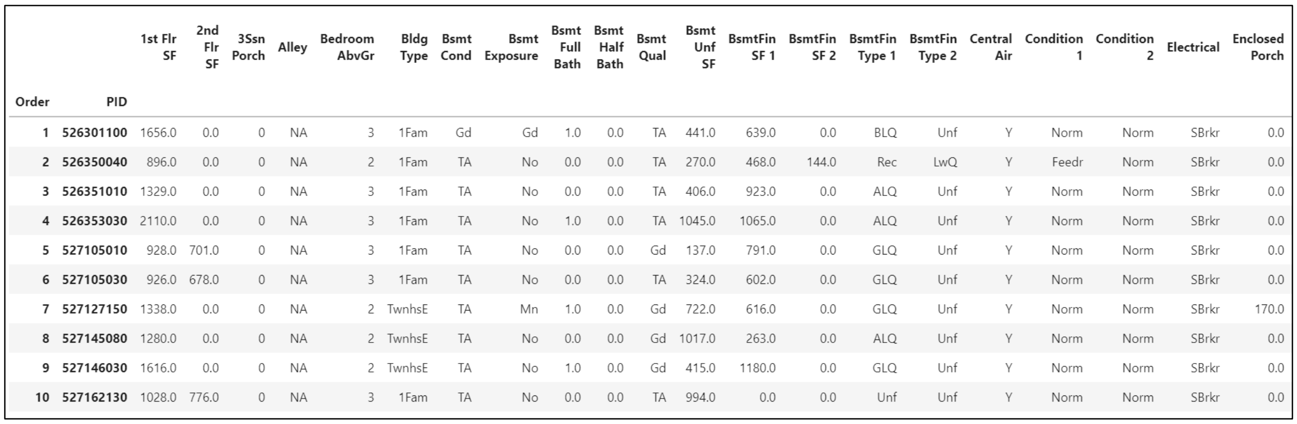
The two paper also predict house prices and shows the comparison of the hedonic price model and artificial neural network, and the second paper shows that multilevel models perform better among the three models. My implementation uses regression models to predict house prices and compares them.

3. Methods

I have referred to a similar implementation of the paper on Kaggle which uses XGBoost, Support Vector Regression, Random Forest, Ridge, Decision Tree, and K-Nearest Neighbors since the exact implementation of the referred paper was not found online.

3.1 Dataset

The House dataset presented by De Cock (2011) is used. The dataset describes the sale of residential units in Ames, Iowa starting from 2006 until 2010. The dataset contains many variables; it contains 2930 records and 82 features https://www.kaggle.com/c/house-prices-advanced-regression-techniques/data



Order	PID	1st Flr SF	2nd Flr SF	3Ssn Porch	Alley	Bedroom AbvGr	Bldg Type	Bsmt Cond	Bsmt Exposure	Bsmt Full Bath	Bsmt Half Bath	Bsmt Qual	Bsmt Unf SF	BsmtFin SF 1	BsmtFin SF 2	BsmtFin Type 1	BsmtFin Type 2	Central Air	Condition 1	Condition 2	Electrical	Enclosed Porch
1	**526301100**	1656.0	0.0	0	NA	3	1Fam	Gd	Gd	1.0	0.0	TA	441.0	639.0	0.0	BLQ	Unf	Y	Norm	Norm	SBrkr	0.0
2	**526350040**	896.0	0.0	0	NA	2	1Fam	TA	No	0.0	0.0	TA	270.0	468.0	144.0	Rec	LwQ	Y	Feedr	Norm	SBrkr	0.0
3	**526351010**	1329.0	0.0	0	NA	3	1Fam	TA	No	0.0	0.0	TA	406.0	923.0	0.0	ALQ	Unf	Y	Norm	Norm	SBrkr	0.0
4	**526353030**	2110.0	0.0	0	NA	3	1Fam	TA	No	1.0	0.0	TA	1045.0	1065.0	0.0	ALQ	Unf	Y	Norm	Norm	SBrkr	0.0
5	**527105010**	928.0	701.0	0	NA	3	1Fam	TA	No	0.0	0.0	Gd	137.0	791.0	0.0	GLQ	Unf	Y	Norm	Norm	SBrkr	0.0
6	**527105030**	926.0	678.0	0	NA	3	1Fam	TA	No	0.0	0.0	TA	324.0	602.0	0.0	GLQ	Unf	Y	Norm	Norm	SBrkr	0.0
7	**527127150**	1338.0	0.0	0	NA	2	TwnhsE	TA	Mn	1.0	0.0	Gd	722.0	616.0	0.0	GLQ	Unf	Y	Norm	Norm	SBrkr	170.0
8	**527145080**	1280.0	0.0	0	NA	2	TwnhsE	TA	No	0.0	0.0	Gd	1017.0	263.0	0.0	ALQ	Unf	Y	Norm	Norm	SBrkr	0.0
9	**527146030**	1616.0	0.0	0	NA	2	TwnhsE	TA	No	1.0	0.0	Gd	415.0	1180.0	0.0	GLQ	Unf	Y	Norm	Norm	SBrkr	0.0
10	**527162130**	1028.0	776.0	0	NA	3	1Fam	TA	No	0.0	0.0	TA	994.0	0.0	0.0	Unf	Unf	Y	Norm	Norm	SBrkr	0.0

Figure 1 Dataset Description

3.2 Methodology

Figure 2 shows the methodology that is followed in the paper.

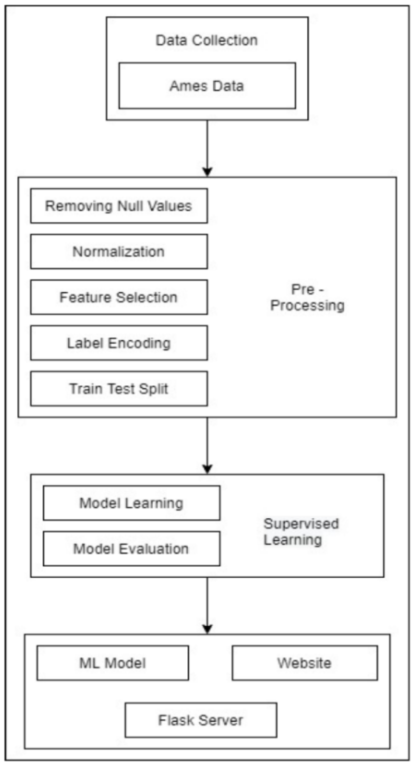
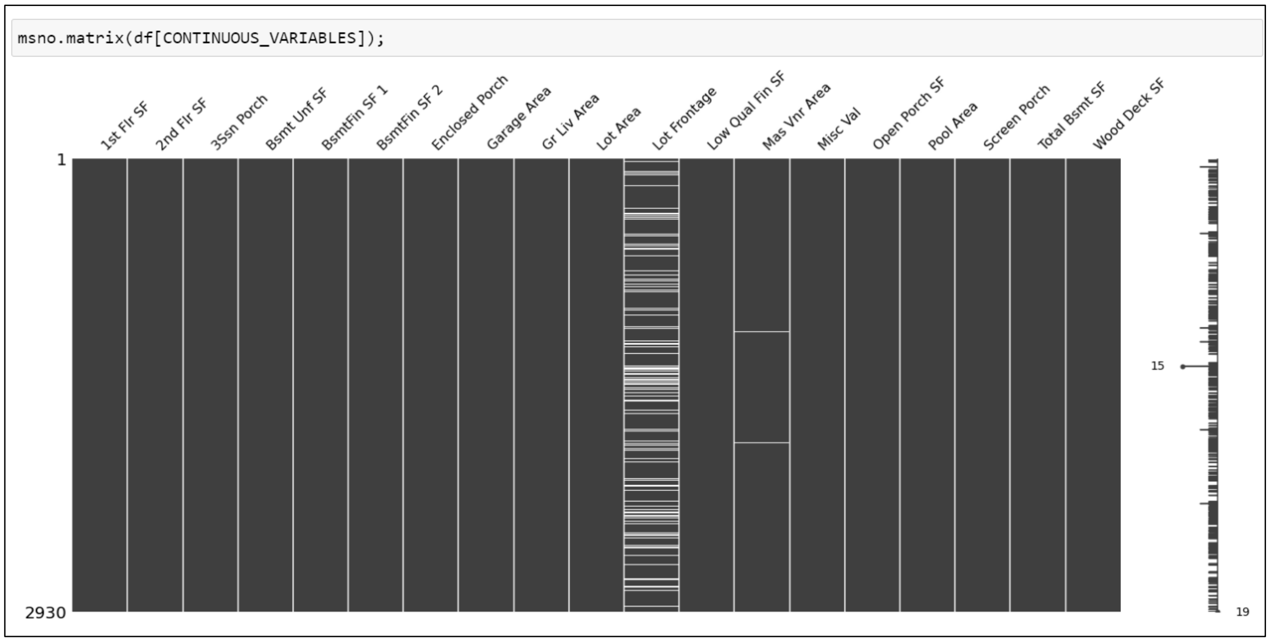


Figure 2 Methodology Applied

3.3 Data Preprocessing

3.3.1 Data Cleaning

Some machine learning models do not accept data with missing values. After checking the missing values in each column, I filled in the missing values with appropriate entries depending on the column. I also visualized the missing data using the 'missingno' library. Figure 3 shows the number of missing values in column 'Lot Frontage', the missing values in the 'Lot Frontage' indicate that the house is not connected to any street, and the missing values are filled with 0.

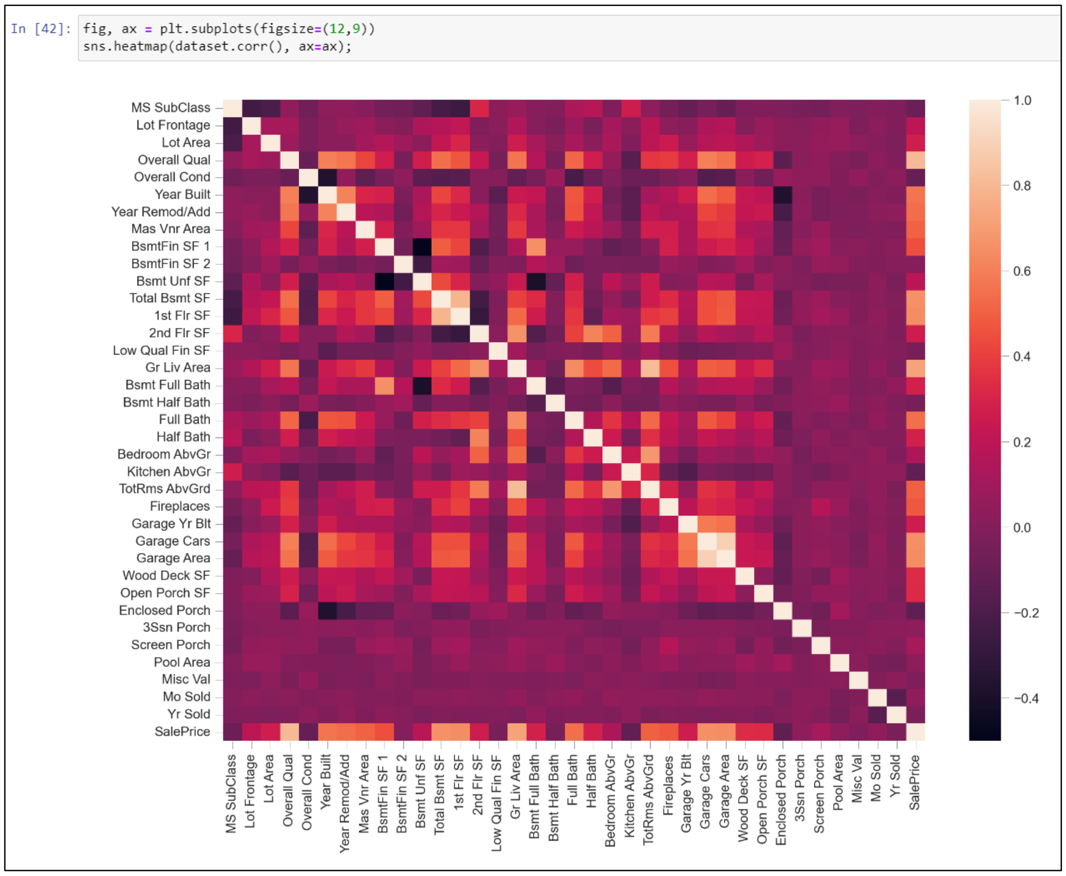


```
msno.matrix(df[CONTINUOUS_VARIABLES]);
```

Figure 3 Missing Values

3.3.2 Data Correlation

Correlation between the dataset variables can be shown using a heatmap graph [Figure 4]. Pearson correlation shows a linear relationship between two variables [Figure 5]. Spearman correlation [Figure 6] shows an ordinal relationship between two variables. The target variable, the Sales price is highly correlated with Overall Quality and Gr Liv Area. The sales price is also highly positively correlated with the Year Built, Year remod/Add, Mas Vnr Area, Total Bsmt SF, and Garage Area.



```
In [42]: fig, ax = plt.subplots(figsize=(12,9))
         sns.heatmap(dataset.corr(), ax=ax);
```

Figure 4 Heatmap showing correlation

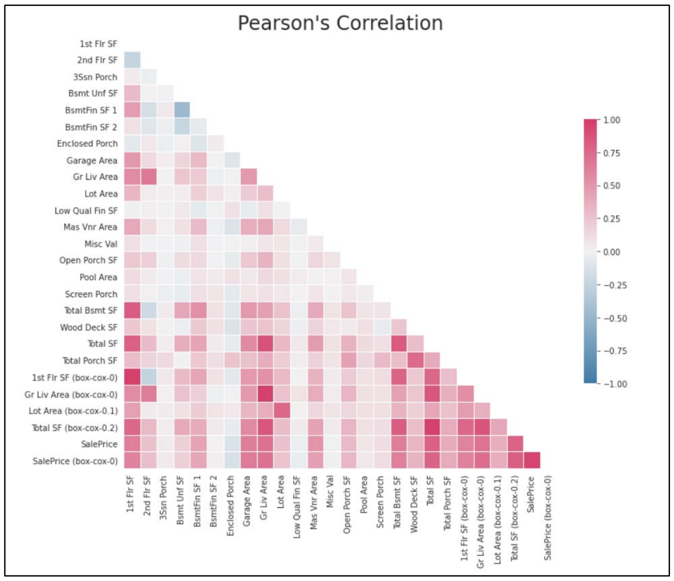


Figure 5 Pearson Correlation

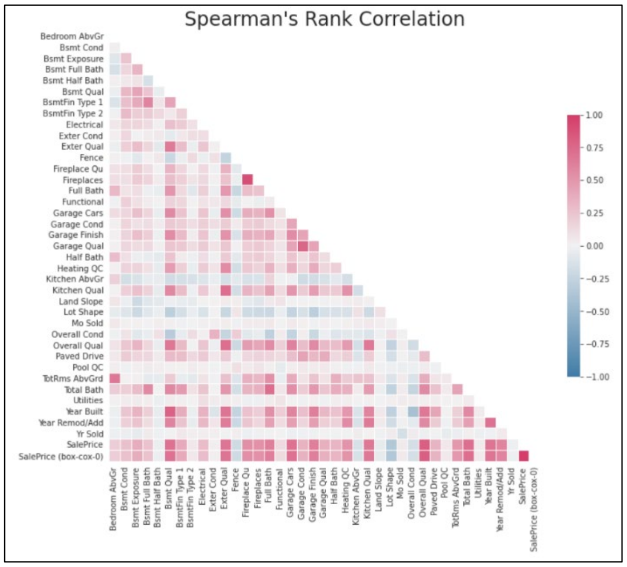


Figure 6 Spearman Correlation

3.3.3 Data Visualization

In this section, I have explored the data using visualizations to understand the data and the relationship between the different features. The main plot used here is Gr Live Area Vs Sales Price as the overall living area is the most correlated predictor.

The main paper has a couple of charts, for instance, the number of Houses Vs Neighborhoods [Figure 8], and Houses built Vs Year [Figure 7].

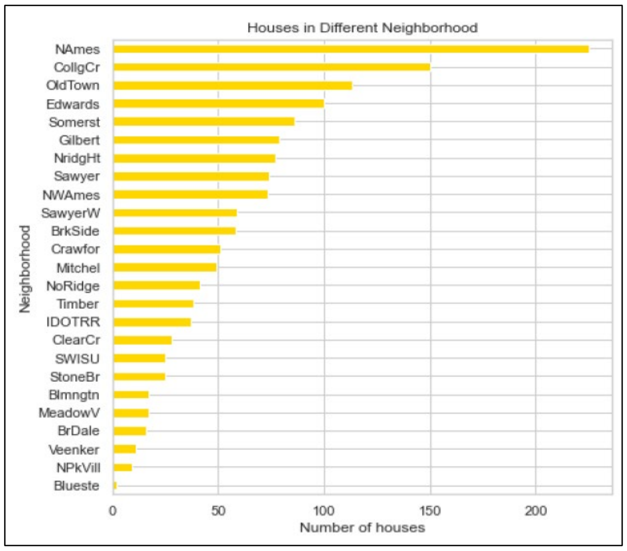


Figure 8 Houses in different neighborhoods

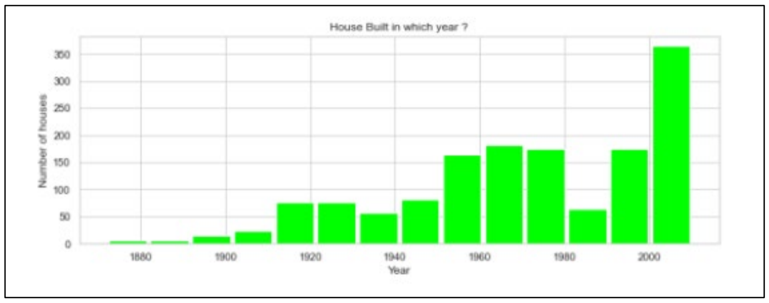
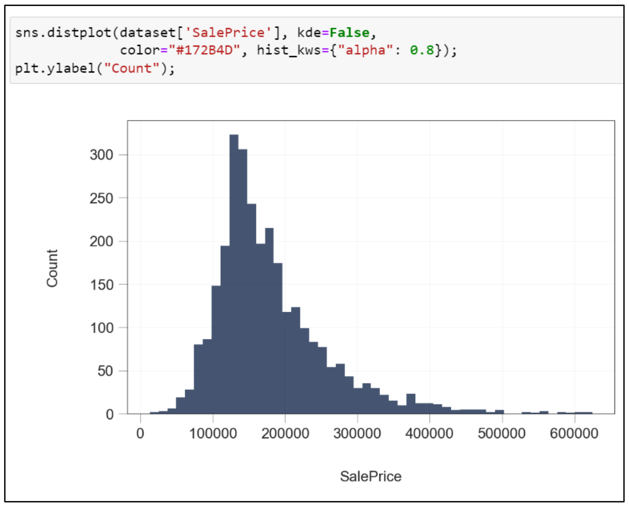


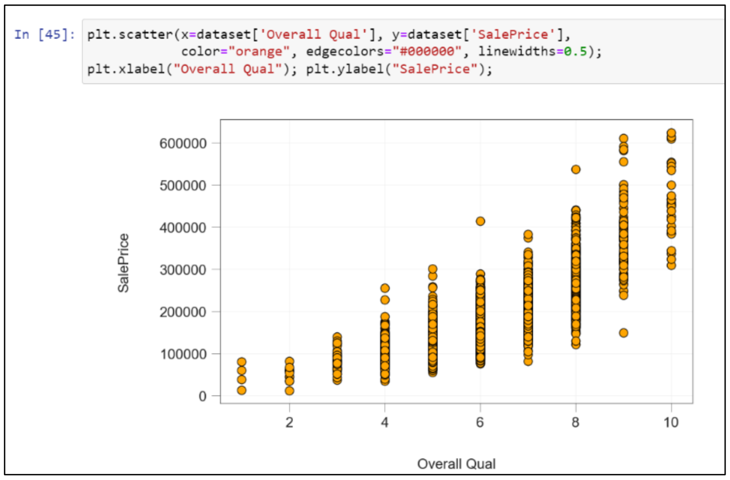
Figure 7 Houses Built Vs Year

Apart from these, some other data visualization that I performed by replicating the different Kaggle notebooks is shown in the following figures. From figure 10, we can see that most house prices fall between 100,000 and 200,000. We also see that there are several expensive houses to the right of the plot. In figure 9, we can see that generally, as the overall quality increases, the sales prices increase too.



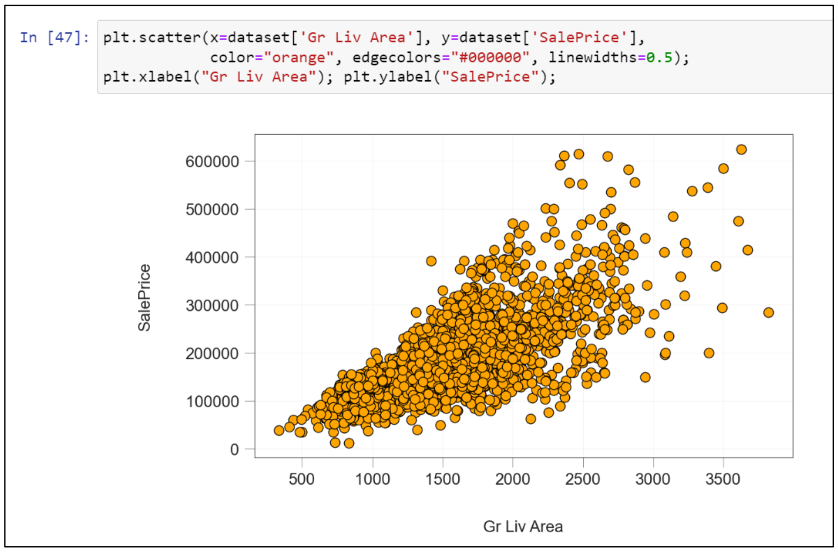
```
sns.distplot(dataset['SalePrice'], kde=False,
             color="#172B4D", hist_kws={"alpha": 0.8});
plt.ylabel("Count");
```

Figure 10 Sale Price vs Count of Houses



```
In [45]: plt.scatter(x=dataset['Overall Qual'], y=dataset['SalePrice'],
                     color="orange", edgecolors="#000000", linewidths=0.5);
         plt.xlabel("Overall Qual"); plt.ylabel("SalePrice");
```

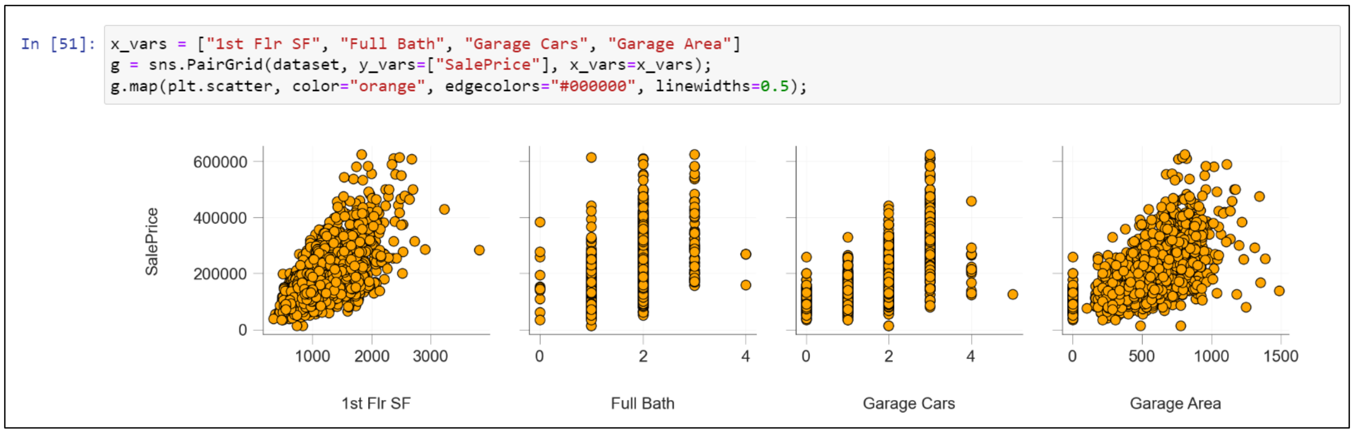
Figure 9 Overall quality Vs Sale Price



```
In [47]: plt.scatter(x=dataset['Gr Liv Area'], y=dataset['SalePrice'],
                     color="orange", edgecolors="#000000", linewidths=0.5);
         plt.xlabel("Gr Liv Area"); plt.ylabel("SalePrice");
```

Figure 11 Gr Living Area Vs Sale Price

The scatter plot in Figure 11 shows clearly that there is a strong correlation between Gr Liv Area and Sales Price. From Figure 12, we can see that these four variables are truly positively correlated with the target variable. However, they are not as correlated as Overall Qual and Gr Liv Area. Figure 13 shows the price distribution by neighborhood.



```
In [51]: x_vars = ["1st Flr SF", "Full Bath", "Garage Cars", "Garage Area"]
         g = sns.PairGrid(dataset, y_vars=["SalePrice"], x_vars=x_vars);
         g.map(plt.scatter, color="orange", edgecolors="#000000", linewidths=0.5);
```

Figure 12 Sale Price Vs other Variables

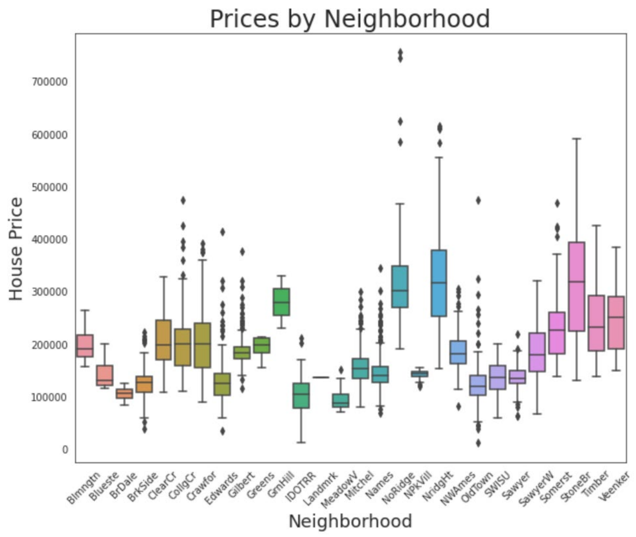


Figure 13 House prices by neighborhood

3.3.4 Outlier Removal

The outliers can be detected using a technique called Isolation Forest. Figure 13 shows the plot of the Gr Liv area with the sale price. The data points that have a Gr Liv area of less than 4000 are kept, and others are removed.

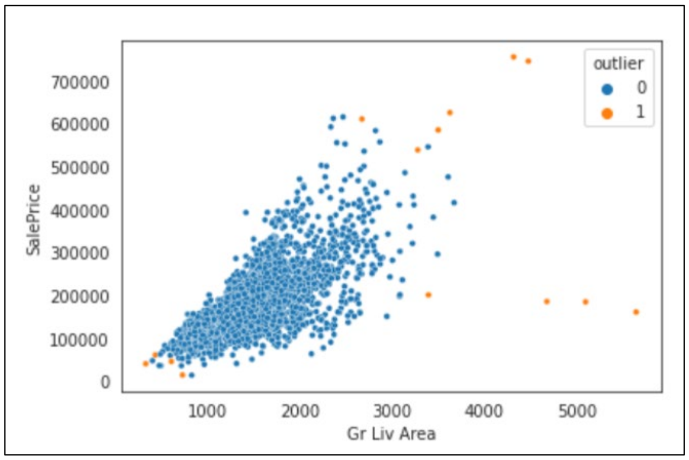


Figure 14 Outliers

3.3.5 Feature Engineering

In this section, the results of the visualizations are used to engineer the features of our dataset. Since machine learning models accept only numbers as input, and since the dataset contains some categorical features, these features are encoded to be suitable for modeling by performing one-hot encoding. Some of the ordinal data that is present in the dataset are mapped to numbers. To avoid the multicollinearity problem, one feature from each pair of highly correlated predictors is deleted.

3.4 Modeling Approach

For each of the techniques used for modeling, the following approach is followed:

- Choose an algorithm that implements the corresponding technique.
- For the algorithm, an effective parameter technique is chosen.
- A model is created using the parameters found.
- On the training dataset, train the model.
- On the model, test the dataset and get the results.

3.5 Modeling Techniques

The sale price that is to be predicted is a continuous value, which means the prediction type for this problem is regression.

1. Linear Regression:

Linear regression is a model that predicts a relationship of direct proportionality between the dependent and the predictor variables that produce a straight line.

2. Support Vector Regression:

Support vector regression is a method of fitting regression to data using the idea of support vector and Lagrangian multiplier. Support vector regression is the promotion of support vector machines in regression problems.

3. K Nearest Neighbors

In this technique, the model tries to find the number(k) of training examples closest in distance to a new point and predict the output for this new point from this closest neighbor.

4. Decision Technique

In this technique, the target variable is predicted by learning simple decision rules inferred from the data features.

5. Random Forest

Bagging is an ensemble method where many base models are used with a randomized subset of the data to reduce the variance of the base model

6. Gradient Boosting

Random Forest regression uses an ensemble learning approach for regression. It is a method that combines the predictions from multiple machine learning models to make an accurate prediction.

3.5 Performance Metric:

Mean Absolute Error (MAE) is used for evaluating the model.

$$\mathrm{MAE}(y, \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| .$$

4. Experiments:

This section shows the experiments performed on the dataset using the above-mentioned approaches.

4.1 Linear Regression

Ridge regression and Elastic Net are used as implementations for linear regression.

4.1.1 Ridge Regression

1. Searching best parameters using GridSearchCV()

```
In [66]: from sklearn.model_selection import GridSearchCV
         from sklearn.linear_model import Ridge

         parameter_space = {
             "alpha": [1, 10, 100, 290, 500],
             "fit_intercept": [True, False],
             "solver": ['svd', 'cholesky', 'lsqr', 'sparse_cg', 'sag', 'saga'],
         }

         clf = GridSearchCV(Ridge(random_state=3), parameter_space, n_jobs=4,
                            cv=3, scoring="neg_mean_absolute_error")

         clf.fit(X_train, y_train)
         print("Best parameters:")
         print(clf.best_params_)

         Best parameters:
         {'alpha': 290, 'fit_intercept': True, 'solver': 'cholesky'}
```

2. Building the model using the best parameters found using GridSearchCV()

```
In [67]: ridge_model = Ridge(random_state=3, **clf.best_params_)
```

3. Training the model

```
In [68]: ridge_model.fit(X_train, y_train);
```

4. Evaluating the model performance

```
In [69]: from sklearn.metrics import mean_absolute_error

         y_pred = ridge_model.predict(X_test)
         ridge_mae = mean_absolute_error(y_test, y_pred)
         print("Ridge MAE =", ridge_mae)

         Ridge MAE = 15270.463549642736
```

4.1.2 Elastic Net

1. Searching best parameters using GridSearchCV()

```
In [70]: from sklearn.linear_model import ElasticNet

         parameter_space = {
             "alpha": [1, 10, 100, 280, 500],
             "l1_ratio": [0.5, 1],
             "fit_intercept": [True, False],
         }

         clf = GridSearchCV(ElasticNet(random_state=3), parameter_space,
                            n_jobs=4, cv=3, scoring="neg_mean_absolute_error")

         clf.fit(X_train, y_train)
         print("Best parameters:")
         print(clf.best_params_)

         Best parameters:
         {'alpha': 280, 'fit_intercept': True, 'l1_ratio': 1}
```

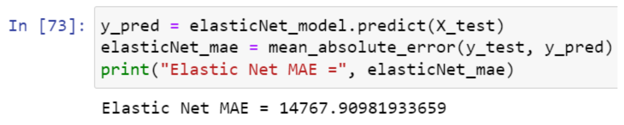
2. Building the model using the best parameters using GridSearchCV()

```
In [71]: elasticNet_model = ElasticNet(random_state=3, **clf.best_params_)
```

3. Training the model

```
In [72]: elasticNet_model.fit(X_train, y_train);
```

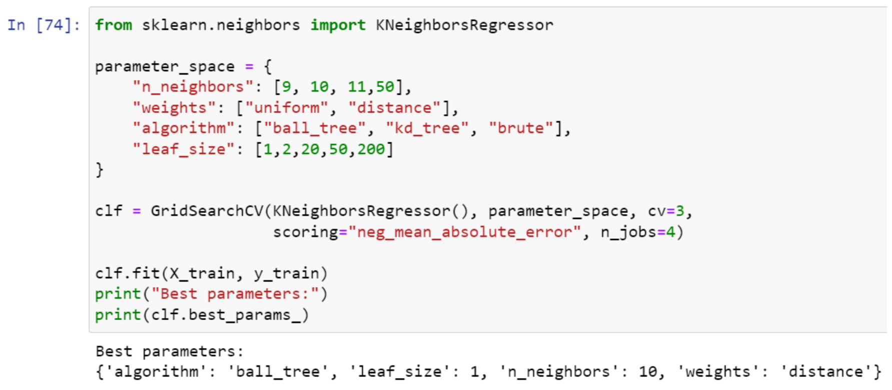
4. Evaluating the model



```
In [73]: y_pred = elasticNet_model.predict(X_test)
         elasticNet_mae = mean_absolute_error(y_test, y_pred)
         print("Elastic Net MAE =", elasticNet_mae)

         Elastic Net MAE = 14767.90981933659
```

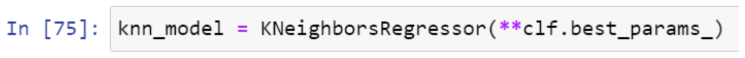
4.2 K Nearest Neighbors

1. Searching best parameters using GridSearchCV()



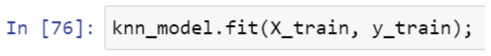
```
In [74]: from sklearn.neighbors import KNeighborsRegressor

         parameter_space = {
             "n_neighbors": [9, 10, 11,50],
             "weights": ["uniform", "distance"],
             "algorithm": ["ball_tree", "kd_tree", "brute"],
             "leaf_size": [1,2,20,50,200]
         }

         clf = GridSearchCV(KNeighborsRegressor(), parameter_space, cv=3,
                            scoring="neg_mean_absolute_error", n_jobs=4)

         clf.fit(X_train, y_train)
         print("Best parameters:")
         print(clf.best_params_)

         Best parameters:
         {'algorithm': 'ball_tree', 'leaf_size': 1, 'n_neighbors': 10, 'weights': 'distance'}
```

2. Building the model using the best parameters using GridSearchCV()



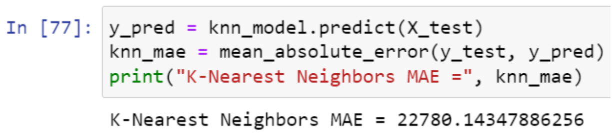
```
In [75]: knn_model = KNeighborsRegressor(**clf.best_params_)
```

3. Training the model



```
In [76]: knn_model.fit(X_train, y_train);
```

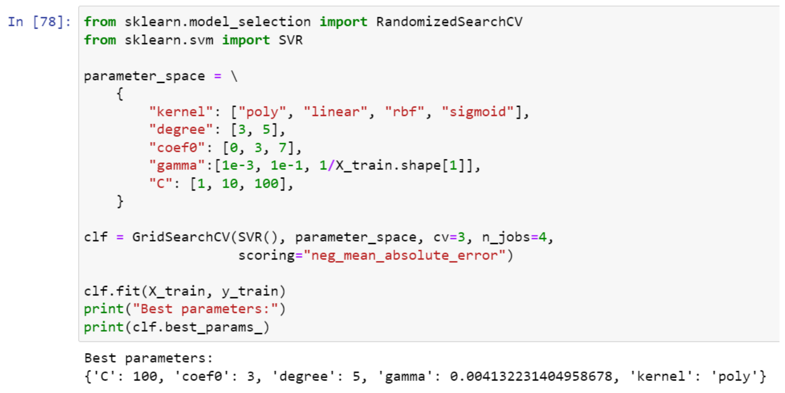
4. Evaluating the model



```
In [77]: y_pred = knn_model.predict(X_test)
         knn_mae = mean_absolute_error(y_test, y_pred)
         print("K-Nearest Neighbors MAE =", knn_mae)

         K-Nearest Neighbors MAE = 22780.14347886256
```

4.3 Support Vector Regression

1. Searching best parameters using GridSearchCV()



```
In [78]: from sklearn.model_selection import RandomizedSearchCV
         from sklearn.svm import SVR

         parameter_space = \
             {
                 "kernel": ["poly", "linear", "rbf", "sigmoid"],
                 "degree": [3, 5],
                 "coef0": [0, 3, 7],
                 "gamma":[1e-3, 1e-1, 1/X_train.shape[1]],
                 "C": [1, 10, 100],
             }

         clf = GridSearchCV(SVR(), parameter_space, cv=3, n_jobs=4,
                            scoring="neg_mean_absolute_error")

         clf.fit(X_train, y_train)
         print("Best parameters:")
         print(clf.best_params_)

         Best parameters:
         {'C': 100, 'coef0': 3, 'degree': 5, 'gamma': 0.004132231404958678, 'kernel': 'poly'}
```

2. Building the model using the best parameters using GridSearchCV()

```
In [79]: svr_model = SVR(**clf.best_params_)
```

3. Training the model

```
In [80]: svr_model.fit(X_train, y_train);
```

4. Evaluating the model

```
In [81]: y_pred = svr_model.predict(X_test)
         svr_mae = mean_absolute_error(y_test, y_pred)
         print("Support Vector Regression MAE =", svr_mae)

         Support Vector Regression MAE = 12874.927869351326
```

4.4 Random Forest

1. Searching best parameters using GridSearchCV()

```
In [90]: from sklearn.ensemble import RandomForestRegressor

         parameter_space = \
             {
                 "n_estimators": [10, 100, 300, 600],
                 "criterion": ["mse", "mae"],
                 "max_depth": [7, 50, 254],
                 "min_samples_split": [2, 5],
                 "min_samples_leaf": [1, 5],
                 "max_features": [19, 100, X_train.shape[1]],
                 "bootstrap": [True, False],
             }

         clf = RandomizedSearchCV(RandomForestRegressor(random_state=3),
                                  parameter_space, cv=3, n_jobs=4,
                                  scoring="neg_mean_absolute_error",
                                  n_iter=10, random_state=3)

         clf.fit(X_train, y_train)
         print("Best parameters:")
         print(clf.best_params_)

         Best parameters:
         {'n_estimators': 600, 'min_samples_split': 2, 'min_samples_leaf': 1, 'max_features': 19, 'max_depth': 254, 'criterion': 'mse',
         'bootstrap': False}
```

2. Building the model using the best parameters using GridSearchCV()

```
In [91]: rf_model = RandomForestRegressor(**clf.best_params_)
```

3. Training the model

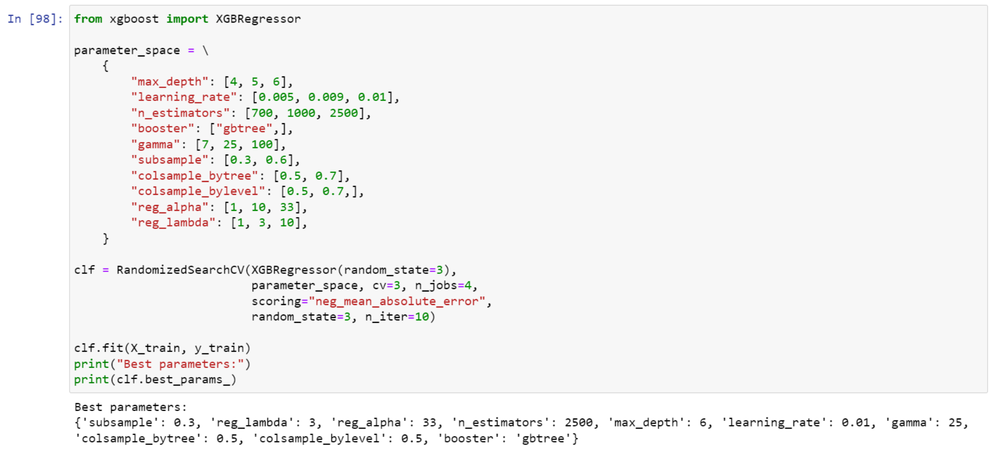
```
In [92]: rf_model.fit(X_train, y_train);
```

4. Evaluating the model

```
In [93]: y_pred = rf_model.predict(X_test)
         rf_mae = mean_absolute_error(y_test, y_pred)
         print("Random Forest MAE =", rf_mae)

         Random Forest MAE = 14435.075557832422
```

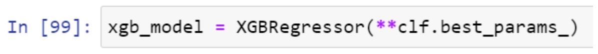
4.4 Gradient Boosting

1. Searching best parameters using GridSearchCV()



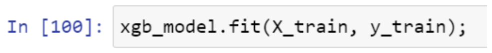
```
In [98]: from xgboost import XGBRegressor

         parameter_space = \
             {
                 "max_depth": [4, 5, 6],
                 "learning_rate": [0.005, 0.009, 0.01],
                 "n_estimators": [700, 1000, 2500],
                 "booster": ["gbtree",],
                 "gamma": [7, 25, 100],
                 "subsample": [0.3, 0.6],
                 "colsample_bytree": [0.5, 0.7],
                 "colsample_bylevel": [0.5, 0.7,],
                 "reg_alpha": [1, 10, 33],
                 "reg_lambda": [1, 3, 10],
             }

         clf = RandomizedSearchCV(XGBRegressor(random_state=3),
                                  parameter_space, cv=3, n_jobs=4,
                                  scoring="neg_mean_absolute_error",
                                  random_state=3, n_iter=10)

         clf.fit(X_train, y_train)
         print("Best parameters:")
         print(clf.best_params_)

         Best parameters:
         {'subsample': 0.3, 'reg_lambda': 3, 'reg_alpha': 33, 'n_estimators': 2500, 'max_depth': 6, 'learning_rate': 0.01, 'gamma': 25,
         'colsample_bytree': 0.5, 'colsample_bylevel': 0.5, 'booster': 'gbtree'}
```

2. Building the model using the best parameters using GridSearchCV()



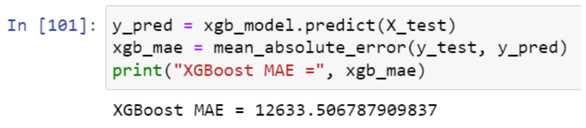
```
In [99]: xgb_model = XGBRegressor(**clf.best_params_)
```

3. Training the model



```
In [100]: xgb_model.fit(X_train, y_train);
```

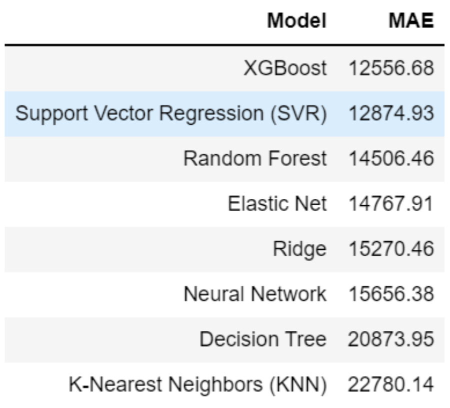
3. Evaluating the model



```
In [101]: y_pred = xgb_model.predict(X_test)
          xgb_mae = mean_absolute_error(y_test, y_pred)
          print("XGBoost MAE =", xgb_mae)

          XGBoost MAE = 12633.506787909837
```

5. Results

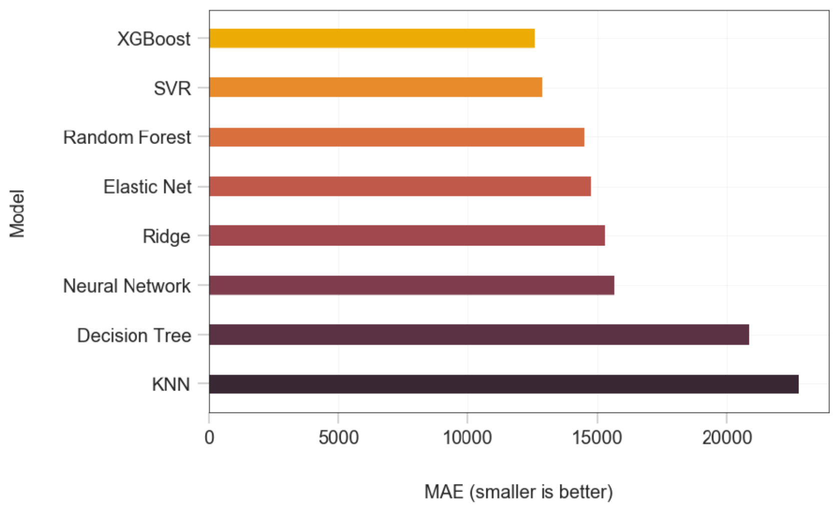
The following table shows the MAE scores for each of the models, MAE scores sorted in ascending order.



Model	MAE
XGBoost	12556.68
Support Vector Regression (SVR)	12874.93
Random Forest	14506.46
Elastic Net	14767.91
Ridge	15270.46
Neural Network	15656.38
Decision Tree	20873.95
K-Nearest Neighbors (KNN)	22780.14

The following graph visualizes the table of Models and MAE scores.

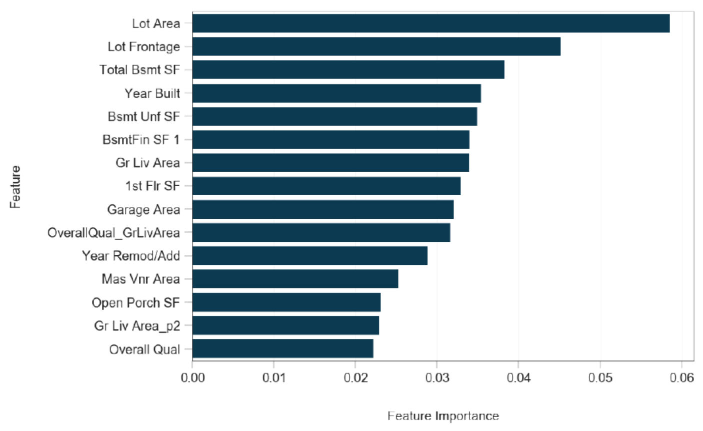
```
In [102]: x = ['KNN', 'Decision Tree', 'Neural Network', 'Ridge',
               'Elastic Net', 'Random Forest', 'SVR', 'XGBoost']
          y = [22780.14, 20873.95, 15656.38, 15270.46, 14767.91,
               14506.46, 12874.93, 12556.68]
          colors = ["#392834", "#5a3244", "#7e3c4d", "#a1484f",
                    "#c05949", "#d86f3d", "#e88b2b", "#edab06"]
          fig, ax = plt.subplots()
          plt.barh(y=range(len(x)), tick_label=x, width=y, height=0.4, color=colors);
          ax.set(xlabel="MAE (smaller is better)", ylabel="Model");
```



In this experiment, the best model is XGBoost and the worst model is K-Nearest Neighbors. The difference between the MAE score of the best model and the worst model is significant. MAE presents a value that is easy to understand; it shows the average value of model error. For example, for our XGBoost model, its MAE is 12556.68 which means that on average, XGBoost will predict a value that is bigger or smaller than the true value by 12556.68.

Some important features determined by the XGBoost model are shown in the below figure:

```
xgb_feature_importances = xgb_model.feature_importances_
xgb_feature_importances = pd.Series(
    xgb_feature_importances, index=X_train.columns.values
    ).sort_values(ascending=False).head(15)

fig, ax = plt.subplots(figsize=(7, 5))
sns.barplot(x=xgb_feature_importances,
            y=xgb_feature_importances.index,
            color="#003f5c");
plt.xlabel('Feature Importance');
plt.ylabel('Feature');
```



6. Conclusion

In this study, I proposed comparing machine learning models to predict house prices. I replicated the model and showed the data science process of getting the data, then cleaning and preprocessing the data, followed by exploring the data and building models, then evaluating the results, and showing them with visualizations. From this project, it can be concluded that the XGBoost model has the maximum efficiency and has higher predicting power compared to the other models.

7. My Contributions

After getting the dataset from Kaggle, I researched various research papers and studied and understood my main paper which compared different regression models. I was able to replicate and implement the project by following this Kaggle project [5]. Apart from this project, I also studied another Kaggle project [6] which showed that manually 'improved' data outperformed the data that was only cleaned with minimum effort.

8. References

[1] De Cock, D. (2011). Ames, Iowa: Alternative to the Boston housing data as an end-of-semester regression project. Journal of Statistics Education, 19(3)

[2] Feng, Y., & Jones, K. (2015, July). Comparing multilevel modeling and artificial neural net-works in house price prediction. In Spatial Data Mining and Geographical Knowledge Services.

[3] Limsombunchai, Visit & Gan, Christopher & Lee, Minsoo. (2004). House Price Prediction: Hedonic Price Model vs. Artificial Neural Network. American Journal of Applied Sciences.

[4] Kangane, Pranav & Mallya, Aadesh (2021). Analysis of Different Regression Models for Real Estate Price Prediction. International Journal of Engineering Applied Sciences and Technology

[5] https://www.kaggle.com/code/ammar111/house-price-prediction-an-end-to-end-ml-project

[6] https://github.com/webartifex/ames-housing

[7] Alfityatin, Adyan (2017) Modeling houses Price Prediction using Regression Analysis and Particle Swarm Optimization. International Journal of Advanced Computer Science and Applications.