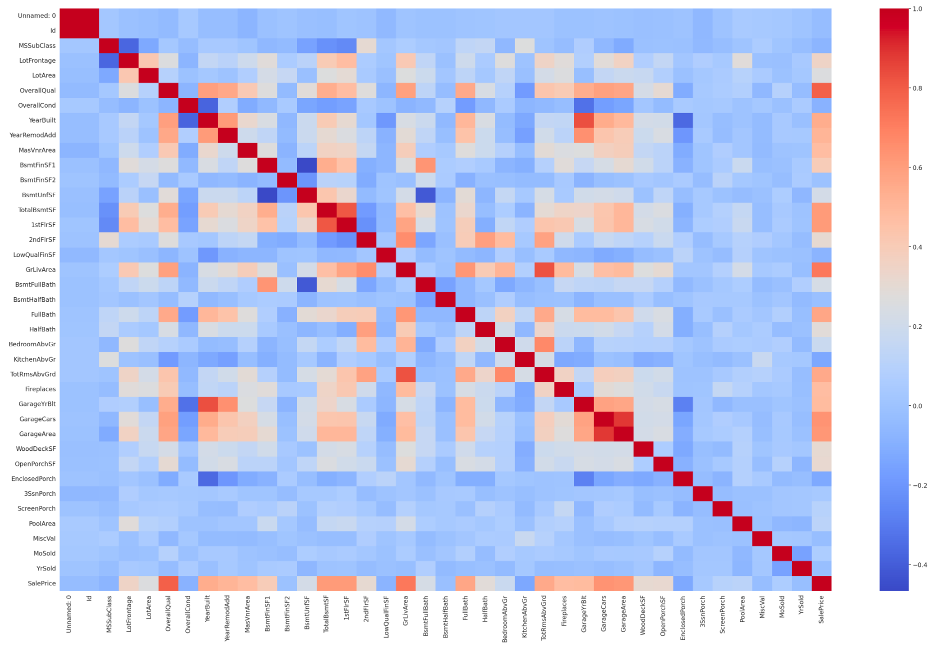
ML Project Group 4

The training data process must be explicitly explained.

The dataset we have is a house price prediction dataset of regression type. The target variable ‘Sale Price’ is a numerical value. The dataset includes 82 rows, which needs to be reduced down to 35 features by incorporating various processes like feature engineering, feature selection, and dimensionality reduction. The training dataset contains 1022 rows and 82 columns including the target column.

We will first of all produce a correlation matrix showing the correlation among each features of the dataset. The aim is to identify the columns having high correlation with each other, so that we can remove some of those highly correlated columns and reduce the features of our column. We found out that '1stFlrSF', 'GarageArea', 'GarageYrBlt', 'Id' & 'TotRmsAbvGrd' are having higher than 80% correlation. We decided to keep the First Floor square feet feature, GarageArea & Total rooms above grade, whereas removed the GarageYrBlt and Id column. The Id column is just a randomly generated numeric value hence removed. GarageYrBlt is having time series data, which is not quite useful as we already have the year in which house was build or renovated. We kept First floor square feet as it’s useful in setting the price, area of the house is quite important for the prediction. Garage Area column is also to be removed as another column, contains information regarding the number of cars accommodated by a particular house.



Moving forward let’s see the number of null values in our dataset. We found out the percentage of missing values in each column and decided to remove the columns having more than half of data as null. Thus we removed the `PoolQC` ,`MiscFeature` , `Alley` , `Fence` and

`FireplaceQu` column from our dataframe. The above mentioned columns were having more than 47% of data as null, thus we can drop those columns from our dataset, instead of imputing huge amount of values. Thus we are left with 75 columns. On close examination we found out that `GarageType`, `GarageFinish` , `GarageQual` , `GarageCond` were having missing values. So we imputed the None as the data in all those cells of dataframe, assuming there is no information available. For the columns `LotFrontage` and `MasVnrArea`, we used a smart technique to impute their data. As we know that a neighborhood have some type of symmetry in the structure of the houses. So we will impute `LotFrontage` i.e. linear feet of street connected to property according to the median value of that particular neighborhood on which that house is located. To do that we have used the groupby function, and filled the missing values with the median values of the neighborhood where the house is located. Similarly we imputed `MasVnrArea` value after grouping by the `MasVnrType` column. Thus we are left with 75 columns.

Let's begin feature engineering. We can modify the features related to the construction of the house such as the year in which the house was constructed, the year in which the house was sold, the year in which it was remodeled i.e. renovation if any. As the years and all is a time series based data, we need to do something and turn it into meaningful feature. A newly renovated house have generally higher selling prices, hence we will add that column as feature in our dataset. If the year of building of house and remodel year are different, the house is said to be renovated. Next we have introduced `houseAge` as a feature by taking into account `YrSold` and `YearRemodAdd` . Thus we have now 74 columns. Next we have introduced count of number of bathrooms in house in `totBathrooms` column. Similarly combining the Porch Area of the houses into `totPorchSqft` column. Columns are now reduced to 67.Now introducing a new column `totSqrFt` by combining `BsmtFinSF1` ,`BsmtFinSF2` ,`2ndFlrSF` and because the area is the most influential factor while determining the price of a commodity. To further reduce the columns we have combined `Heating` and `HeatingQC` to form `heatingQualCond.` Further let's transform the categorical columns into numerical columns like `TotalBsmtSF` into `HasBsmt` and `Fireplaces` into `HasFirePlace`. Some skewed columns like `MiscVal` , `LowQualFinSF & PoolArea` are to be dropped. The columns has now reduced to 61. Next we have replaced the categorical values `Neighborhood` ,`Exterior1st` ,`Exterior2nd` and `heatingQualCond` .

Now we will use feature importance given by various models like Random Forest and Linear regression to figure out the top most influential features of our dataset. To perform that we have one-hot encoded the categorical columns into numerical ones, then evaluated on testing dataset. Then we have used excel to find the most important 35 features out of the above 61 mentioned columns of our dataset. To find the importance, we have sorted the numerical and categorical columns in excel according to the higher values of their importance. For one-hot encoded columns we have combined the importance of all values of one column and then ranked it such that top most 35 features are found out of all those 61 features. Thus finally, the dataset is being reduced to the most significant 35 features.

Evaluate the train models

After splitting the dataset into `X_final_train`,`y_train`, `X_final_test & y_test`, let’s find the accuracy metrics of our models i.e. MSE value on Linear Regression and Random Forest. Initially the value of MSE for Linear Regression on default parameters is 960104859.235705 and 870550474.50 on Random Forest. We have used models like `RandomForestRegressor` ,`LinearRegression` ,`SVR` , `AdaBoostRegressor, LGBMRegressor, XGBRegressor` and `GradientBoostingRegressor` . Now we need to find the best parameters for our all the models. We will use Kfold cross validation and grid search cv in order to find the best parameters for the machine learning algorithms to be used. We have passed the training dataset into the function model_evaluate_gridsearch() which is passed inside the pen_ultimate_function. The pen_ultimate_function includes model and parameters grid as input. The inner function model_evaluate_gridsearch will take the subgroups of dataset of validation and training and then run the main function on it while taking in the parameter mentioned or passed inside it. The k is taken to be 5, thus the 5 subgroups are formed from the training dataset. The training dataset is learned by each model and each particular hyperparameter using grid search cv and tested via mean square accuracy scoring parameter. At the end the function returns the best parameter found out using grid search, thus we get the hyperparameter which predicts the least mean sq error. The data is splitted into training and validation set initially. Then each subgroups are passed inside the gridsearchcv function and fitted and predicted, returning the mean square error for each iteration. Finally after passing in the parameters and grid and model name, our pen_ultimate_function will return the best parameter found on each of the sub group along with the metric like mse,rmse, etc.

Following hyperparameters were found out to be the best parameter by grid search,
`AdaBoostRegressor(learning_rate= 0.5, loss= "linear", n_estimators= 100), RandomForestRegressor(max_depth= 30, max_features= "sqrt", min_samples_leaf= 1, min_samples_split= 2, n_estimators= 200), SVR(C= 10, gamma= "scale", kernel= "rbf"),, GradientBoostingRegressor(learning_rate= 0.1, max_depth= 4, min_samples_leaf= 1, min_samples_split= 2, n_estimators= 200), LGBMRegressor(colsample_bytree= 0.8, learning_rate= 0.1, max_depth= 4, min_child_samples= 20, n_estimators= 100, num_leaves= 31, subsample= 0.8) and XGBRegressor(colsample_bytree= 0.8, learning_rate= 0.1, max_depth= 4, min_child_weight= 1, n_estimators= 200, subsample= 1.0).`

Adaboost builds a model by combining multiple weak models and it focuses on the mistakes made by those models. Random forest builds an ensemble of decision trees where each tree is trained on a small sample of dataset. SVR finds a plane which represents a particular relationship between input and output features. GradientBoosting also builds a strong model by adding weak models. LGBM Regressor is tree based function, aims on improving the speed and efficiency, thus making it useful for the larger dataset. XGBoost Regressor is known for it's performance and regularization techniques. It uses advance regularization techniques to prevent overfitting.

Evaluate the test predictions

After getting the best parameters on each of the model we will run the algorithms again on the best parameter founded. After testing individually each model will be placed into stack and stackingRegressor will be runned after passing each model with best tuned hyperparameter. The final MSE value on the stacking method is found out to be `661650286.`