Saniya Naphade(spn5272)

Final Report: House Prices - Advanced Regression Techniques

Introduction:

Purchasing a house is one of the biggest and most expensive decision of a person's life. A lot of forethought goes into it and various factors are to be considered. It is a difficult decision to make as one would like to buy a house at the best price possible with minimal risk and would like it to be the best investment.
This project, helps buyers understand how various features of their dream house impact the price of the house. It helps get a holistic idea about how height of the ceiling, number of bedrooms or even having white picket fence influences the price of the house.
The dataset is collected from the residential homes in Ames, Iowa. It provides 79 exploratory features for about 1460 houses in the training set and 1458 in the testing set. Given the various features of the house our end goal is to predict the price of the house taking all the features and their influence into consideration.
Of the 79 features, there are 36 numerical variables with 43 variables of object type.

Method:

The best performance obtained using ensemble of XGBRegressor(), GradientboostingRegressor(), LGBMRegressor() and SVR()

On testing dataset:

RMSE : 0.118

The Best result is obtained for ensemble model with the following parameters:
XGBRegressor with the following parameters.

learning_rate	0.05
max_depth	3
colsample_bytree	0.3
n_estimator	1250

The hyperparameter tuning done using GridSearchCV.

parameters = {'learning_rate': [0.02,0.03, 0.05, 1],

'max_depth': [3,4,5],
'colsample_bytree': [0.3, 0.4, 0.5, 0.6],
'n_estimators': [1000,1250, 1500, 1800]}

give the range of values tried for various parameters.

GradientboostingRegressor with the following parameters:

learning_rate	0.05
loss	'huber'
Max_depth	3
n_estimator	500

The hyperparameter tuning done using GridSearchCV.

parameters = {'learning_rate': [0.02, 0.05, 0.06],

'max_depth': [3, 4, 5],

'n_estimators': [500, 1000]}

give the range of values tried for various parameters.

LGBMRegressor with the following parameters:

learning_rate	0.005
max_bin	100
n_estimator	500
num_iterations	10000
num_leaves	25

The hyperparameter tuning done using GridSearchCV.

parameters = {'learning_rate': [0.005, 0.05, 0.1],
'max_bin': [100, 200],
'n_estimators': [500, 1000]
'num_iterations':[5000,10000],
'num_leaves': [20,25,30]}

give the range of values tried for various parameters.

SVR with the following parameters:

C	1
gamma	0.01
epsilon	0.005

The hyperparameter tuning done using GridSearchCV:

parameters = {'C': [0.1, 1, 100], 'epsilon': [100, 200],
'gamma':[0.01,1,100]}
gives the range of values tried for various parameters.

The ensemble model with equal contributions like 25% contribution from each of the four models performed the best.

Feature Engineering and Selection:

Remove outliers for GrLivArea feature.
Looking at the correlation matrix, I removed some of the features which were not highly correlated with the dependant variable (SalesPrice). Features removed:
{'GarageYrBlt','TotRmsAbvGrd','1stFlrSF','GarageArea'}

I also engineered some new features to better predict the target variable. The engineered features further improved the performance of the model.
Logtransformed SalePrice as it was skewed.

Engineered features:

1. Added feature 'lastupdate' which gives the time frame of how long after the house was built was it remodeled.
2. Feature 'extra_features' is obtained by taking a product of the YearBuilt feature with FirePlace feature
3. Feature 'value' is calculated as a product of the YearBuilt and the OverallQual of the house.
4. Feature 'garage_value' is the product of the YearBuilt and GarageCars of the house.

5. New feature 'area' helped understand the relation between the independent variables, LotArea and LotFrontage, and the dependent variable, SalesPrice.

Data Preprocessing:

https://scikit-learn.org/stable/auto_examples/inspection/plot_permutation_importance.html #sphx-glr-auto-examples-inspection-plot-permutation-importance-py

Used this scikit example as a reference to construct pipeline for numerical and categorical preprocessing. The difference between this method and the method implemented in the project is, for numerical processing, I have added StandardScaler() method to standardize the data.
I tried Label Encoder as one of the means for categorical encoding, however, the RMSE score obtained was higher than the one obtained using one hot encoding.

Other methods tried:

1. Method (1):

Detecting Outliers:
In method, I dealt with the outliers that were present in the training data. I used Tukey method to find the indices with outliers for each column and then used Counter to find how many columns for a given index had outliers depending on their distance from the center (median). Then removed indices with more that 3 counts. I tried with various cleaning parameters and observed that cleaning parameter of 1.5 gives the best RMSE score.

Feature Engineering:
In this checkpoint I added quite a few features along with the existing features as mentioned in the best method.

1. I performed ranking among categories for certain columns with respect to the median of SalePrice - target variable.
2. I added a few more newly engineering features. I used https://developers.google.com/machine-learning/crash-course/feature-crosses/video-lecture, for developing new engineering features.
3. Used correlation matrix to determine which features(both new and existing ones) to keep and which to remove. In the end add a total of 106 features
4. Performed sqrt for variables but it showed no significant improvement in the distribution of the variables.
5. Log transformation of only SalePrice and GridLivArea showed some improvement.

Optimization of Hyperparameters:
Used optuna library to optimize the hyperparameters for XGB, Lasso, Randomforest, SVR, LGBM, and GradientBoost regressors.
Finally used StackingCVRegressor to combine all above optimized regressors at Level1 and used Linear Regressor at level2.

Data Preprocessing:
Same as that used for the best method. Used StandardScalar() to scale the numerical features and used one-hot encoding to encode categorical features.

Following were the optimized hyperparameters:
SVR
Optimized parameters: {'C': 0.4416961498853393, 'epsilon': 0.010073918817564937, 'coef0': 0.999914452085788}

Lasso

Optimized parameters: {'alpha': 0.0002670186195986862}

Random Forest
Optimized parameters: {'n_estimators': 193, 'max_depth': 10, 'min_samples_split': 5, 'min_samples_leaf': 3, 'max_features': 49}

LGBM
Optimized parameters: {'num_leaves': 18, 'max_depth': 7, 'learning_rate': 0.12502640588007136, 'n_estimators': 50, 'min_child_weight': 0.3497230676455781}

XGB
Optimized parameters: {'colsample_bytree': 0, 'max_depth': 4, 'learning_rate': 0.30420100433105085, 'n_estimators': 146, 'min_child_weight': 0.5777264784109637}

GradientBoosting
Optimized parameters: {'max_depth': 3, 'learning_rate': 0.12375620375521373, 'n_estimators': 119 }

Obtained RMSE score of 0.12347
With various features added/removed, the score obtained was always higher than that of the best method. Out of which the best score obtained was using an ensemble(ensemble1) of SVR, XGB, LGBM,and Gradient Boost each at 25% and no outliers removed. Received an RMSE score of 0.11920 considering the same hyperparameters as mentioned above.

2. Method (2):

Step1: Removal of only uncorrelated features. Removed features:
{'Id','LowQualFinSF','YrSold','BsmtFinSF2','MiscVal','BsmtHalfBath'}

Step2:
Data Preprocessing:
Same as that used for the best method. Used StandardScalar() to scale the numerical features and used one-hot encoding to encode categorical features.

Step3: Used XGBRegressor, RandomForestRegressor and SVR separately. Used GridSearch to fine-tune the models.
Final model parameters:
XGBRegressor:

learning_rate	0.02
max_delta_step	0
booster	gbtree
colsample_bytree	0.6
gamma	0.0
min_child_weight	0
max_depth	4
n_estimator	1250
subsample	0.7
tree_method	auto
verbosity	none

RandomForestRegressor:

N_estimator	100
Max_depth	90
Max_features	3
Min_samples_leaf	3
Min_samples_split	8
bootstrap	True

SVR:

kernel	rbf
epsilon	0.001
C	0.1
gamma	1

Results obtained for various methods tried:

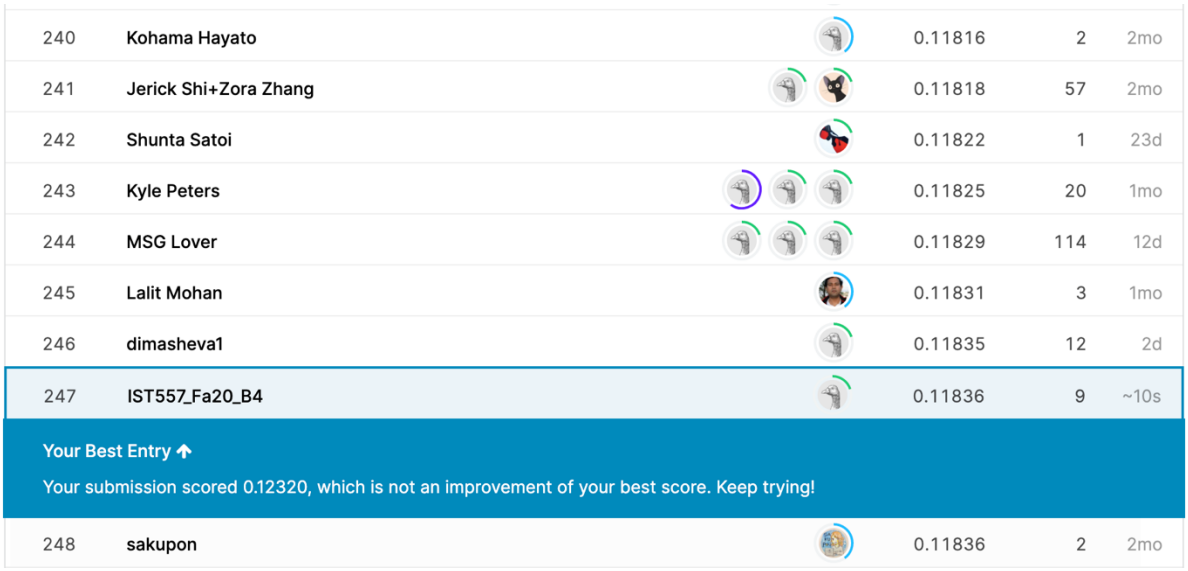
Method	RMSE
Best Method	0.11836
Method(1): 1. StackingRegressor 2. Ensemble(1)	 1. 0.12347 2. 0.11920
Method(2): 1. XGBRegressor 2. SVR 3. RandomForestRegressor	 1. 0.12549 2. 0.3999 3. 0.217297084

Hypothesis for poor results for other methods:

Method (1):
I believe that the performance of the model after removing the outliers and adding new features worsened due to Curse of Dimensionality as I ended up with more features than datapoints.
Adding of new features and removal of the outliers did not help with the model performance.
One of the reasons could be the curse of dimensionality. The other reason I believe is due to high computational requirements, I could not take a very wide range of values for hyperparameter tuning.
Moreover, I am not dealing with the skew that is present in the data which could also be an influencing factor over the poor performance of the model.

Method (2):
There were no new features added plus feature engineering aspect was missing. I believe these factors contributed to poor performance.
There are a lot of aspects missing in this method such as presence of outliers, skew in the data. Improper imputation of missing values.

Screenshot of the best method on leader board:



Summary:

Part1:
Technical Summary:

1. Learnt how to detect and deal with outliers such as Tukey's method. This project also helped me to better analyze graphs which helped me understand how the individual features related with the dependent variable

2. Learnt about feature engineering and generation of new features. https://developers.google.com/machine-learning/crash-course/feature-crosses/video-lecture -This reference helped me generate new, helpful features which helped improve the model performance.

3. Discovered new methods to fine tune models on a larger scale. Optuna is a faster and a more precise method to fine tune models s compared to GridSearch method.

Part2:

During the course of this project, I learnt various techniques can be used to handle data effectively. I learnt about various methods to generate new features and read graphs to understand the relation between variables. Had I know how to handle the various discrepancies such as outliers, skew in the data before I would have been able to implement the ensemble model a little earlier which would have given me enough time to generate new even better features.

References:

[1] https://scikit- learn.org/stable/auto_examples/inspection/plot_permutation_importance.html#sphx-glr-auto-examples-inspection-plot-permutation-importance-py

[2] https://towardsdatascience.com/clean-efficient-data-pipelines-with-pythons-sklearn-2472de04c0ea

[3] https://www.kdnuggets.com/2017/01/3-methods-deal-outliers.html

[4] https://towardsdatascience.com/10-hyperparameter-optimization-frameworks-8bc87bc8b7e3

[5] https://analyticsindiamag.com/stackingcvregressor-in-python/