

A Project Report

on

House Price Prediction using Supervised Learning

*Submitted in partial fulfillment of the
requirement for the award of the degree of*

**Bachelor of Technology in Computer Science and
Engineering**



(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

**Under The Supervision of
Dr. K.M.Baalamurugan
Assistant Professor**

Department of Computer Science and Engineering

Submitted By

18SCSE1180082– SHREYA SINGH

18SCSE1180068-AYUSH MAHESHWARI

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA, INDIA
DECEMBER - 2021**



SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

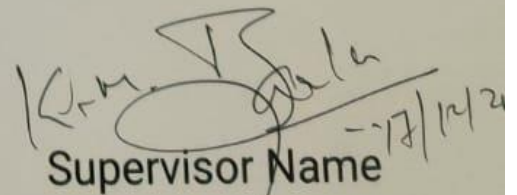
CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled "House Price Prediction using Supervised Learning." in partial fulfillment of the requirements for the award of the BACHELOR OF TECHNOLOGY IN COMPUTER SCIENCE AND ENGINEERING submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of JULY-2021 to DECEMBER-2021, under the supervision of Dr.K.M.Baalamurugan, Assistant Professor, Department of Computer Science and Engineering of Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

18SCSE1180082- SHREYA SINGH
18SCSE1180068-AYUSH MAHESHWARI

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

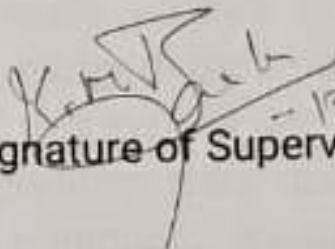

Supervisor Name

Dr. K.M.Baalamurugan
Assistant Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce
examination of 18SCSE1180082 – SHREYA SINGH,
18SCSE1180068 – AYUSH MAHESHWARI has been held on
12/12/21 and his/her work is recommended for the
award of BACHELOR OF TECHNOLOGY IN COMPUTER
SCIENCE AND ENGINEERING.

Signature of Examiner(s)


Signature of Supervisor(s)
- 12/12/21

Signature of Project Coordinator

Signature of Dean

Date: 17/12/2021

Place: Greater Noida

Title: House Price Prediction using Supervised Learning.

ABSTRACT

Machine learning plays a major role from past years in image detection, spam reorganization, normal speech command, product recommendation and medical diagnosis. Present machine learning algorithm helps us in enhancing security alerts, ensuring public safety and improve medical enhancements. Machine learning system also provides better customer service and safer automobile systems. In this we discuss about the prediction of future housing prices that is generated by machine learning algorithm. For the selection of prediction methods, we compare and explore various prediction methods. This study utilizes machine learning algorithms as a research method that develops housing price prediction models. In this project we will be walking you through analyzing the problem from collecting data, importing it to a Jupyter notebook, looking for promising attributes, finding out correlations, plotting graphs, creating a pipeline, dealing with missing values and much more. We will use the model for predicting house prices given a set of features. The concepts we will learn are like cross validation, train-test splitting, stratified shuffle split, cross validation and sampling work in action.

Contents

Title	Page No.
Literature Review	6
Methodology	10
Requirement	24
Diagram	28
Implementation	29
Required Tools	30
Merits	31
Data Description	33
Result	50
Discussion and Conclusion	53
Summary	55
Proof	62

Literature Review

The latest worldwide financial crisis restored a sharp enthusiasm toward both academic and strategy circles on the part of asset costs and specifically lodging costs clinched alongside monetary movement. As Lamer (2007) notes those lodging showcase predicted eight of the ten post globe War ii recessions, acting Concerning illustration An heading woman for those true segment of the economy. Truth be told he dives

Likewise significantly Concerning illustration with state that “Housing is those benefits of the business cycle”. Vargas and silva (2008) contend that lodging costs alterations assume a paramount part in the determination of the stage of the business cycle. There is huge literature writing in regards to U.S. house prices. Rapach Furthermore strauss (2007) use an auto regressive dispersed slack (ARDL) model framework, holding 25 determinants with conjecture genuine lodging cost development to the unique states of the elected Reserve’s eighth region. They discover that ARDL models tend should beat a benchmark AR model. When those economy booms, development and work in the lodging division expand quickly should react should overabundance demand, quickly pushing ostensible house costs upwards. Throughout those withdrawal phase, the drop in private money lessens aggravate

interest Also ostensible house costs. By ostensible house costs normally fall sluggishly since householders would unwilling on bring down their costs. The majority of the conformity will be attained through declines clinched alongside bargains volume bringing about An drop in the development segment and the lodging built vocation. Moreover, Throughout withdrawal and subsidence true house costs fall quickly Likewise general inflationary patterns diminish true house costs much with sticky perceived costs. Recently, a few writers scope to experimental discoveries that house costs can make instrumental molding to determining yield. (Forni etc, 2003; stock and Watson, 2003; Gupta Furthermore Das, 2010; das etc, 2009; 2010; 2011; Gupta and Hartley, 2013). Those lodging development division speaks to an expansive and only aggregate monetary action communicated in the GDP. In recent years, due to the growing trend towards Big Data, machine learning has become a vital prediction approach because it can predict house prices more accurately based on their attributes, regardless of the data from previous years. Several studies explored this problem and proved the capability of the machine learning approach [2],[3],[4]; however, most of them compared the models' performance but did not consider the combination of different machine learning models. S. Lu et al. did conducted an experiment using a hybrid regression technique on

forecasting house price data, but it requires intensive parameter tuning to find the optimal solution [5].

Due to the importance of model combination, this paper adopted the Stacked Generalization approach [6],[7], a machine learning ensemble technique, to optimize the predicted values. We used the “Housing Price in Beijing” dataset, fetched and uploaded to Kaggle by Q. Qiu [8]. By applying several methods on this dataset, we could validate the performance of each individual approach. The lowest Root Mean Squared Logarithmic Error (RMSLE) is 0.16350 on the test set, which belongs to the Stack Generalization method.

Introduction

Commonly, House Price Index (HPI) is used to measure price changes of residential housing in many countries, such as the US Federal Housing Finance Agency HPI, S&P/Case-Shiller price index, UK National Statistics HPI, UK Land Registry’s HPI, UK Halifax HPI, UK Rightmove HPI and Singapore URA HPI. The HPI is a weighted, repeat- sales index, meaning that it measures average price changes in repeat sales or refinancings on the same properties.

housing economists to estimate changes in the rates of mortgage defaults, prepayments, and housing affordability in specific geographic areas [1]. Because HPI is a rough indicator calculated from all transactions, it is inefficient to predict the price of a specific house. Many features such as district, age, and the number of floors also need to be considered instead of just the repeat sales in previous decades. In recent years, due to the growing trend towards Big Data, machine learning has become a vital prediction approach because it can predict house prices more accurately based on their attributes, regardless of the data from previous years. Several studies explored this problem and proved the capability of the machine learning approach [2],[3],[4]; however, most of them compared the models' performance but did not consider the combination of different machine learning models. S. Lu et al. did conducted an experiment using a hybrid regression technique on forecasting house price data, but it requires intensive parameter tuning to find the optimal solution [5].

Due to the importance of model combination, this paper adopted the Stacked Generalization approach [6],[7], a machine learning ensemble technique, to optimize the predicted values. We used the "Housing Price in Beijing" dataset, fetched and uploaded to Kaggle by Q. Qiu [8]. By applying several methods on this dataset, we could validate the performance of each individual approach. The lowest Root Mean Squared Logarithmic Error (RMSLE) is 0.16350 on the test set, which belongs to the Stack Generalization method. The paper is structured as follows: Section 2 illustrates the details of the methodology; Section 3 compares the results; and Section 4 discusses the results, draws a conclusion, as well as proposes further potential directions to study the problem.

Methodology

Data Preprocessing

“Housing Price in Beijing” is a dataset containing more than 300, 000 data with 26 variables representing housing prices traded between 2009 and 2018. These variables, which served as features of the dataset, were then used to predict the average price per square meter of each house.

The next step was to investigate missing data. Variables with more than 50% missing data would be removed from the dataset. The variable “Day on market” was removed because of 157, 977 missing data. Any observation which had missing values were also removed from the dataset. Below are a few feature engineering processes which were done to cleanse the dataset:

Remove attributes indicating the number of kitchens, bathrooms, and drawing rooms due to their ambiguity. Set the number of living rooms (bedrooms were mistranslated to living rooms) in a range from 1 to 4. Add attribute “distance” indicating the distance of the house from the center of Beijing.

Replace attribute “constructionTime” with attribute “age” by deducting the year that the house constructed from the current year (2019).

Set minimum values for attributes “price” and “area”.

Split the attribute “floor” into attributes “floorType” and “floorHeight”. After feature engineering, the dataset was checked for outliers. Through Inter-Quartile Range (IQR), an outlier x

can be detected if:

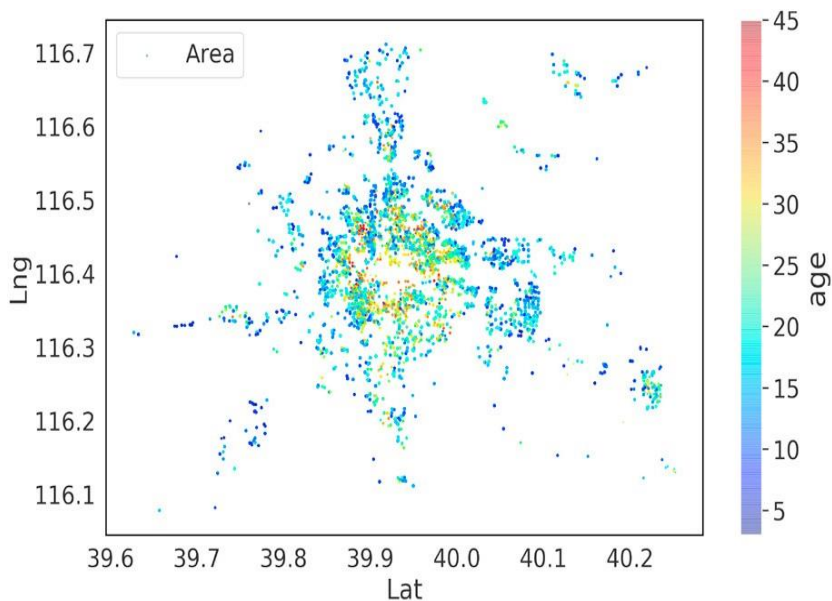
$$x < Q_1 - 1.5 \cdot IQR \quad \text{OR} \quad Q_3 + 1.5 \cdot IQR < x \quad (1)$$

where:

$$Q_1 = 25^{\text{th}} \text{ percentiles} \quad Q_3 = 75^{\text{th}} \text{ percentiles} \quad IQR = Q_3 - Q_1$$

After applying Equation (1) to every column of the dataset, the final dataset contained 231962 data with 19 features, 9 of which were numerical values and 10 of which were categorical values. Table 1 includes details of each attribute.

Fig. 1. Age Distribution



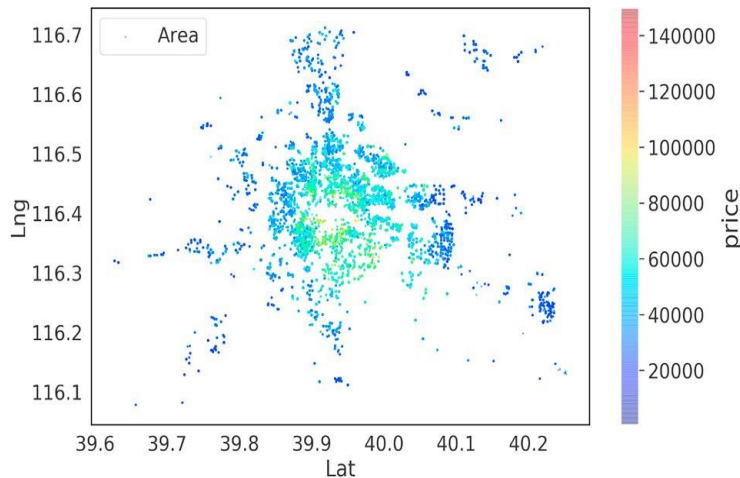


Fig. 2. Price Distribution

Data Analysis

Exploratory data analysis is an essential step before building a regression model. In this way, researchers can discover the implicit patterns of the data, which in turn helps choose appropriate machine learning approaches.

Fig. 1 features houses in Beijing as data points on the map of Beijing. In Fig. 1, the oldest houses are concentrated densely in the center of Beijing, while the newest ones are spread sparsely in the suburban areas. In Fig. 2, the most expensive houses centralizes close to the center of Beijing, while the cheapest ones spreads in the suburban periphery. Since the patterns in Fig. 1 and Fig. 2 are similar, a strong correlation between location, age, and price can be observed. There are also noticeable

differences in housing prices across 13 districts, which are summarized in Fig. 3.

Besides the location features, other features of the house also significantly contribute to the models' performance. In Fig. 4, the difference in price among several building types can be clearly illustrated. Since bungalow is an old building type and is more likely to be built close to the center of Beijing, its price is costly regardless of the small area of a house (Fig. 5).

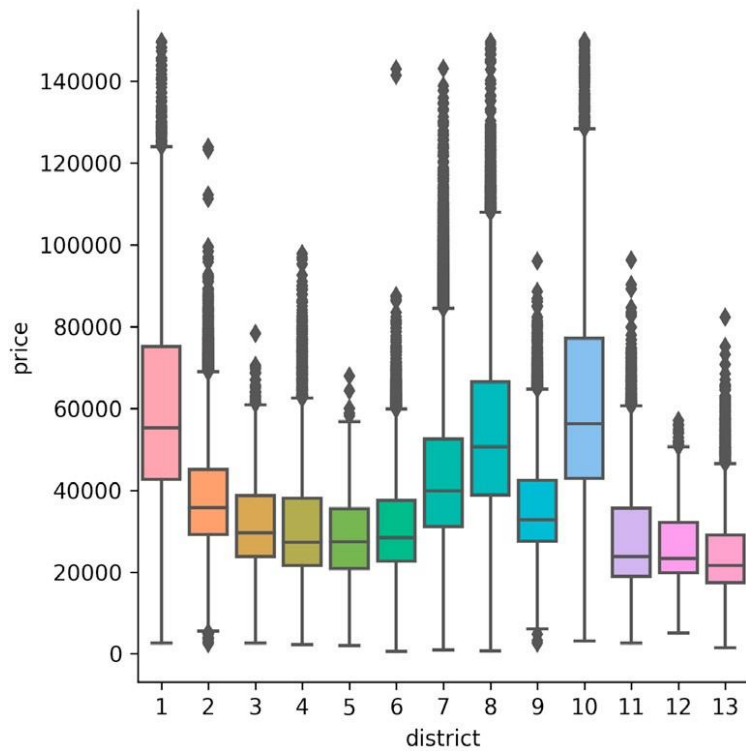


Fig. 3. Correlation between District and Price

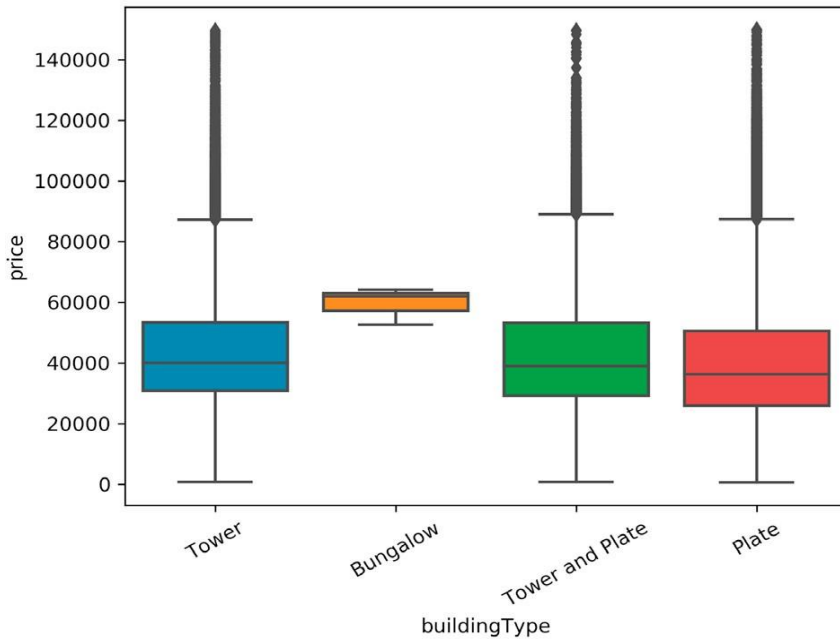


Fig. 4. Correlation between Building Type and Price

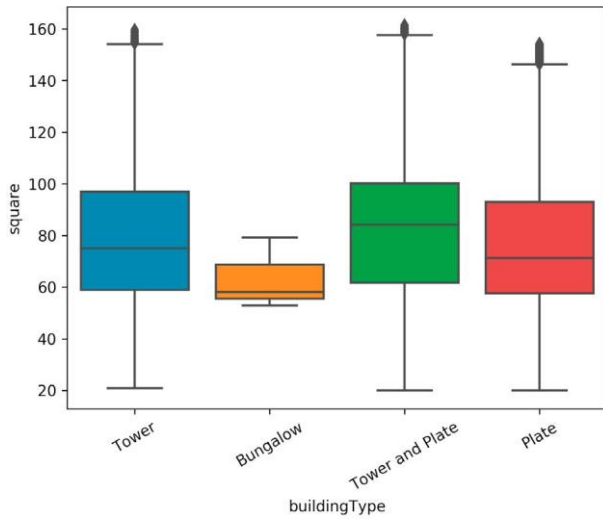


Fig. 5. Correlation between Building Type and Area

Model Selection

Before building models, the data should be processed accordingly so that the models could learn the patterns more efficiently. Specifically, numerical values were standardized, while categorical values were one-hot-encoded. After being processed, the dataset included 58 features. Fig. 6 illustrates the cumulative explained variance. Since most of the features are categorical, it is evident that the variance almost converges at the 30th component. Then, the dataset was split into training set and test set with a ratio of 4 : 1 by utilizing the scikit-learn package [9]. Evaluation function used in this paper is Root Mean Squared Logarithmic Error (RMSLE). This function is illustrated as follow:

$$\text{RMSLE} = \sqrt{\frac{1}{N} \sum_{i=1}^N (\log(y_i + 1) - \log(\hat{y}_i + 1))^2} \quad (2)$$

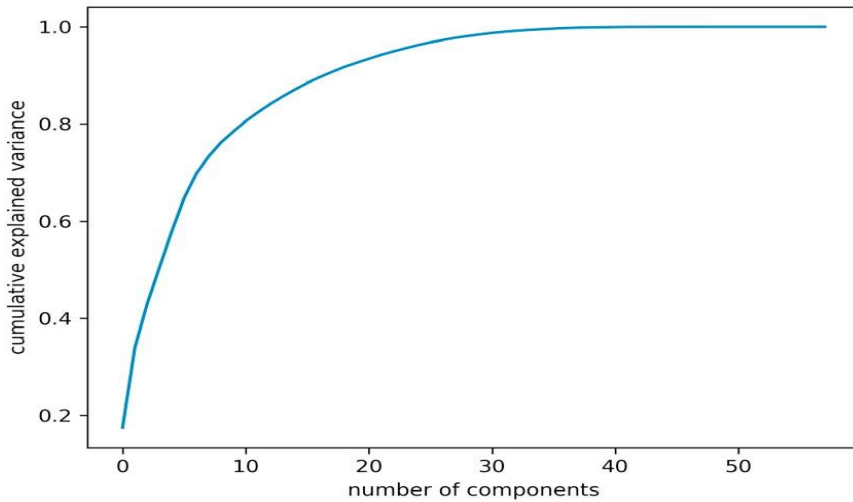


Fig. 6. Cumulative Explained Variance

Random Forest

Random Forest is a kind of ensemble models that combines the prediction of multiple decision trees to create a more accurate final prediction. Random Forest is a verified powerful tool based on previous studies [10]. The random forest algorithm can be summarized in the following steps by S. Raschka and V. Mirjalili in their book “Python Machine Learning” [11]:

Draw a random bootstrap sample of size n (randomly choose n samples from the training set with replacement).

Grow a decision tree from the bootstrap sample, at each node:

Randomly select d features without replacement.

Split the node using the feature that provides the best split according to the objective function, for instance, maximizing the information gain.

Repeat the steps 1-2 k times.

Aggregate the prediction by each tree to assign the class label by majority vote.

In this paper, we used the `RandomForestClassifier` class provided by `sklearn`. The `RandomForestClassifier` has a `n_estimators` parameter that allows to indicate how many trees to build, which we set at 900. While adding more trees to the random forest normally improves accuracy, it also increases the overall training time of the model. The class also includes the `bootstrap` parameter which we set to `True`.

Regarding random forest subsets, however, only a constrained set of features will be used to introduce variation into the trees. By iterating the model multiple times, we also added a few parameters when we initialized the `RandomForestClassifier` to further enhance the performance:

Set `max_depth = 20`, which restricts these trees can only go to 20.

Set `min_samples_split = 10`, which only allows a node that should have at most ten rows before it can be split.

The result of this model on training data is exceptional, 0.12980. The RMSLE of the training set is the lowest among other methods. Fig. 7 illustrates the performance of this model on the training set where the \

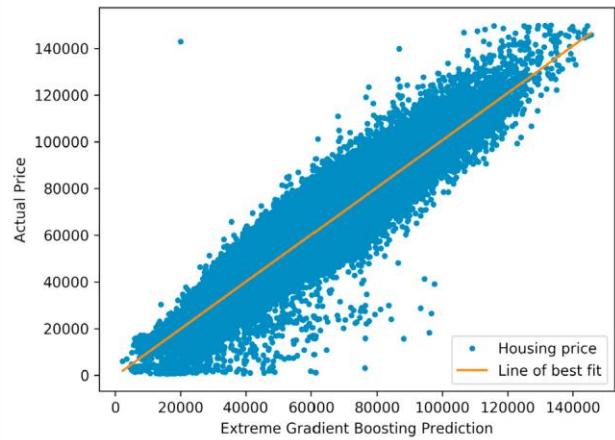
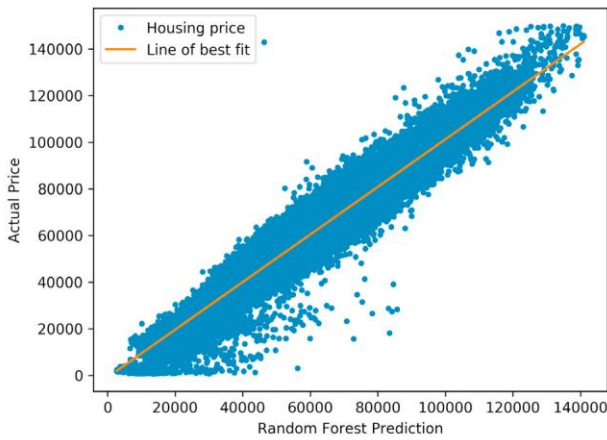


Fig. 7. Random Forest for Training Data Fig. 8. Extreme Gradient Boosting for Training Data

Extreme Gradient Boosting (XGBoost)

XGBoost is a scalable machine learning system for tree boosting. The system is available as an open-source package. The system has generated a significant impact and been widely recognized in various machine learning and data mining challenges [12].

The most crucial reason why XGBoost succeeds is its scalability in all scenarios. The system runs more than ten times faster than existing popular solutions on a single machine and scales to billions of examples in distributed or memory-limited settings. The scalability of XGBoost is due to several major systems and algorithmic optimizations including a novel tree learning algorithm for handling sparse data and a theoretically justified weighted quantile sketch procedure enabling instance weight handling in approximate tree learning. Parallel and distributed computing make learning faster, which allows quicker model exploration. More importantly, the model exploits out-of-core computation and enables

data scientists to process a hundred millions of examples on a desktop. Finally, after combining these techniques to make an end-to-end system, it can scale to even more extensive data with the least amount of cluster resources [12]. In this paper, we utilized the XGBRegressor from xgboost open-source package [13]. After tweaking the XGBoost model multiple times, we set our parameter to the following:

Set `learning_rate = 0.1`

Set `n_estimators = 200`

Determined the optimal tree specific parameters `min_child_weight = 2`, `subsample = 1`, `colsample_bytree = 0.8`

Set regularization parameter: `reg_lambda = 0.45`, `reg_alpha = 0`, `gamma = 0.5`

The model performed with a high accuracy where the RMSLE of the training set is around 0.16118. Fig. 8 shows the Extreme Gradient Boosting prediction in X-axis, and the actual price in Y-axis for training data.

Light Gradient Boosting Machine (LightGBM)

LightGBM is a gradient boosting framework that uses a tree-based learning algorithm. LightGBM has faster training speed with lower memory usage compare to XGBoost [14]. Moreover, it can also support parallel and GPU learning or handle large scale of data. Even though both of the aforementioned boosting models follow Gradient Boosting Algorithm, XGBoost grows tree level-wise and LightGBM grows tree leaf-wise. There are also detailed differences in modelling making them outstanding among different gradient boosting models. In this

paper, we utilized the LGBMRegressor from lightgbm open-source package [15]. The optimal parameters for the model are listed as follows:

Set `learning_rate = 0.15`

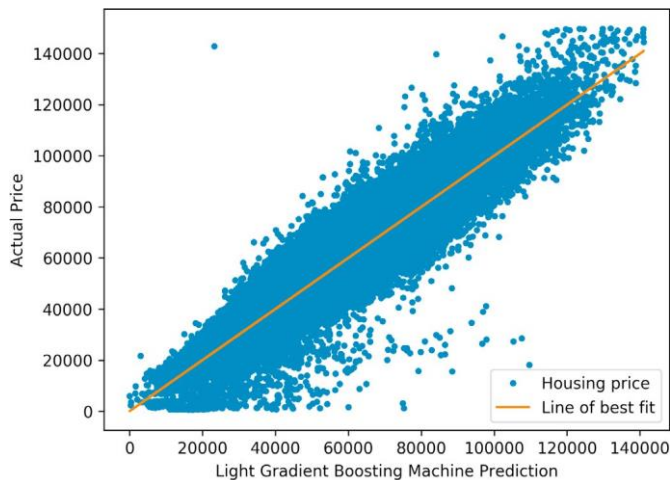


Fig. 9. Light Gradient Boosting Machine for Training Data

Set $n_estimators = 64$

Set optimal tree specific parameters: $min_child_weight = 2$,
 $num_leaves = 36$, $colsample_bytree = 0.8$

Set regularization parameter: $reg_lambda = 0.40$

When applying the LightGBM method, the model achieved a decent accuracy with the loss of 0.16687 on the training set. Even though the performance was not as good as the other methods, LightGBM is the fastest among all models researched on this paper. Fig. 9 illustrates the performance of this model on the training set where the X-axis is the prediction result and the Y-axis is the actual housing price.

Hybrid Regression

Hybrid Regression Model is a model that includes two or more different regression models. The coupling effect of multiple regression models is proven by S. Lu et al., in which a combination of 65% Lasso and 35% XGBoost achieves a RMSLE of 0.11260 on test set of Kaggle [5].

This result is significantly better than results of Lasso and XGBoost in that paper, which are 0.11449 and 0.11843 respectively [5].

A similar approach was also tested on the “Housing Price of Beijing” dataset. The model consisted of 33.33% Random Forest, 33.33% XGBoost, and 33.33% LightGBM from the previous models. The hybrid regression model achieved 0.14969 on the training set. This method averages the individual predictions to form a final prediction. Therefore, it is a fair approach to balance between bias and variance in the composite models. This technique also supports weight assignment to each component model, but it may lead to bias over one model, losing the benefits of generalization. Fig. 10 illustrates the performance of this model on the training set where the X-axis is the prediction result and the Y-axis is the actual housing price.

Stacked Generalization

Stacked Generalization is a machine learning ensembling technique introduced by D. Wolpert [7]. A Python package for stacking named “Vecstack” was built and uploaded to GitHub by I. Ivanov [6]. The main idea of this method is to use the predictions of previous models as features for another model. This approach also utilizes the k-fold cross-validation technique to avoid overfitting.

This paper used the most common architecture, 2-level stacking architecture, to predict the housing price, where the first stacking level consisted of Random Forest and LightGBM and the second stacking level was XGBoost. The stacking model also used 5-fold cross-validation because the dataset is relatively large. The result of the model on the training set, which is 0.16404, is not as impressive as the Hybrid Regression approach. Fig. 11 shows the prediction of the Stacked

Generalization Model on the X-axis and the actual price on the Y-axis.

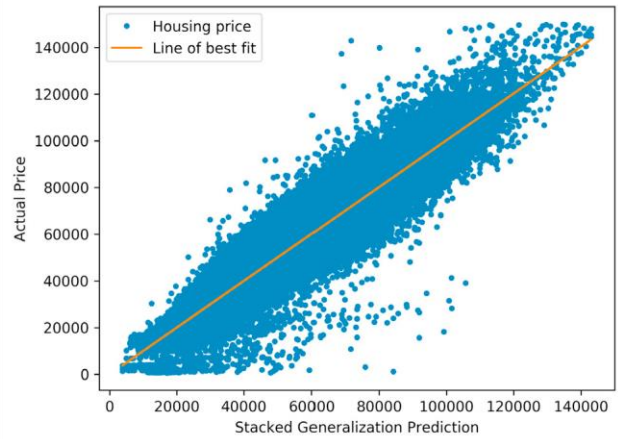
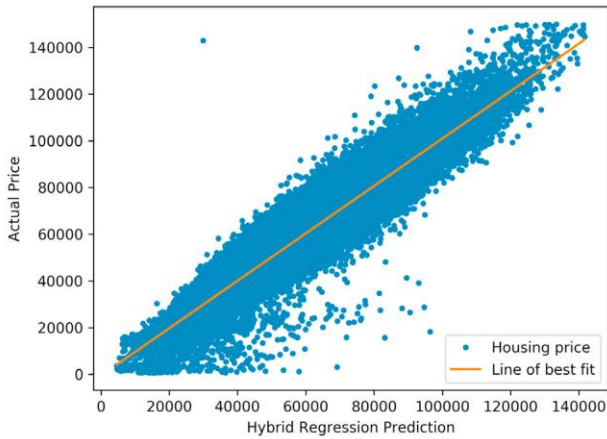


Fig. 10. Hybrid Regression for Training Data

Fig. 11. Stacked Generalization Regression for Training Data

Requirement:

apturl==0.5.2

beautifulsoup4==4.4.1

bleach==2.0.0

blinker==1.3

Brlapi==0.6.4

certifi==2017.4.17

chardet==3.0.3

checkbox-support==0.22

command-not-found==0.3

cryptography==1.2.3

cycler==0.10.0

decorator==4.0.11

defer==1.0.6

docopt==0.6.2

entrypoints==0.2.2

feedparser==5.1.3

freeze==1.0.10

guacamole==0.9.2

html5lib==0.999999999

httplib2==0.9.1

idna==2.5

ipykernel==4.6.1

ipython==6.1.0

ipython-genutils==0.2.0

ipywidgets==6.0.0

jedi==0.10.2

Jinja2==2.9.6

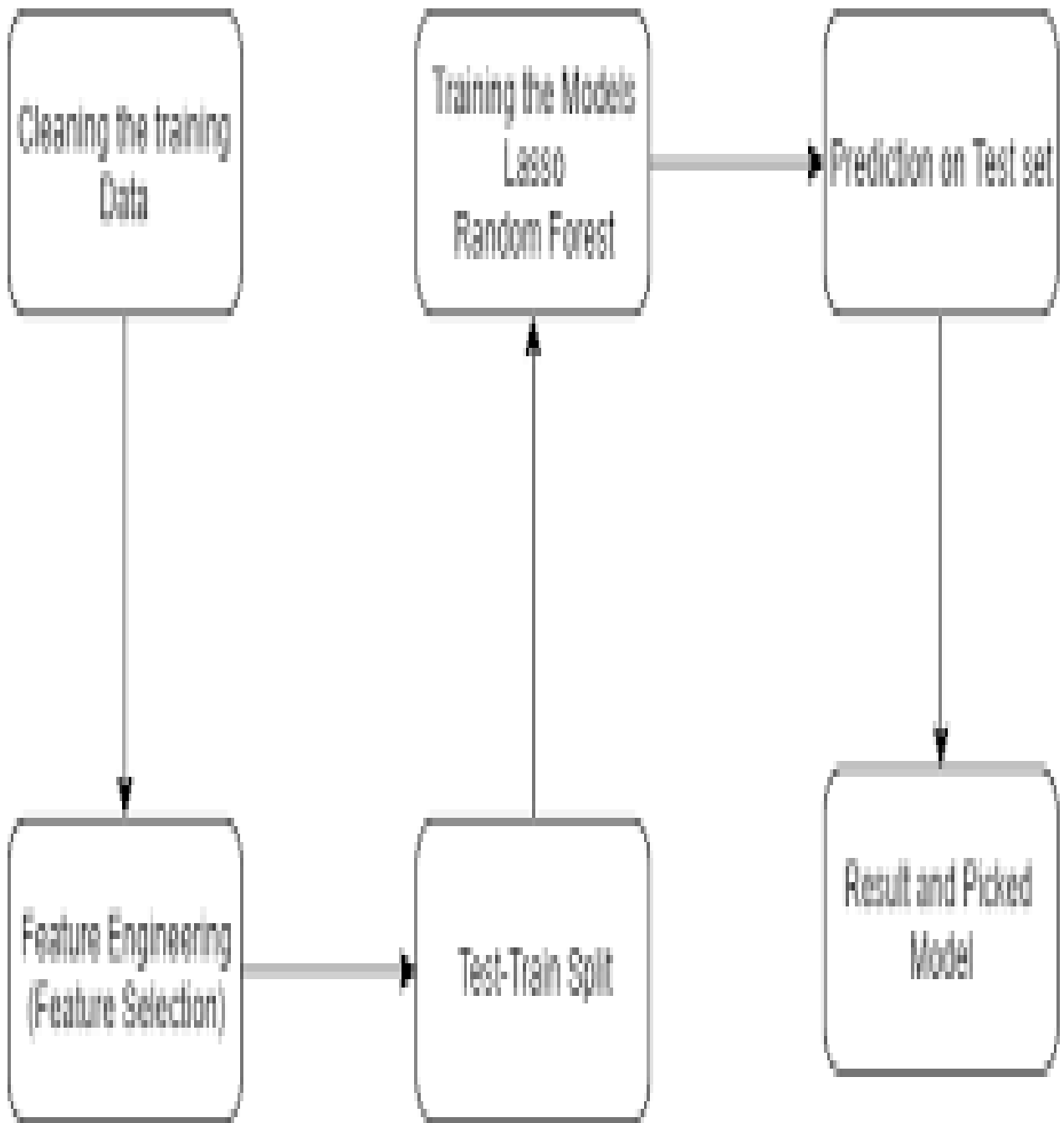
jsonschema==2.6.0

jupyter==1.0.0
jupyter-client==5.0.1
jupyter-console==5.1.0
jupyter-core==4.3.0
language-selector==0.1
louis==2.6.4
lxml==3.5.0
Mako==1.0.3
MarkupSafe==1.0
matplotlib==2.0.2
missingno==0.3.5
mistune==0.7.4
nbconvert==5.2.1
nbformat==4.3.0
notebook==5.4.1
numpy==1.12.1
oauthlib==1.0.3
onboard==1.2.0
padme==1.1.1
pandas==0.20.1
pandocfilters==1.4.1
pexpect==4.2.1
pickleshare==0.7.4
Pillow==3.1.2
pipreqs==0.4.7
plainbox==0.25
prompt-toolkit==1.0.14
protobuf==3.3.0
ptyprocess==0.5.1
pyasn1==0.1.9
pycups==1.9.73
pycurl==7.43.0

Pygments==2.2.0
pygobject==3.20.0
PyJWT==1.3.0
pyparsing==2.2.0
python-apt==1.1.0b1
python-dateutil==2.6.0
python-debian==0.1.27
python-systemd==231
pytz==2017.2
pyxidg==0.25
pyzmq==16.0.2
qtconsole==4.3.0
reportlab==3.3.0
requests==2.17.3
scikit-learn==0.18.1
scipy==0.19.0
seaborn==0.7.1
sessioninstaller==0.0.0
simplegeneric==0.8.1
six==1.10.0
sklearn==0.0
ssh-import-id==5.5
system-service==0.3
tensorflow==1.1.0
terminado==0.6
testpath==0.3.1
tornado==4.5.1
traitlets==4.3.2
ubuntu-drivers-common==0.0.0
ufw==0.35
unattended-upgrades==0.1
unity-scope-calculator==0.1

unity-scope-chromiumbookmarks==0.1
unity-scope-colourlovers==0.1
unity-scope-devhelp==0.1
unity-scope-firefoxbookmarks==0.1
unity-scope-gdrive==0.7
unity-scope-manpages==0.1
unity-scope-openclipart==0.1
unity-scope-texdoc==0.1
unity-scope-tomboy==0.1
unity-scope-virtualbox==0.1
unity-scope-yelp==0.1
unity-scope-zotero==0.1
urllib3==1.21.1
usb-creator==0.3.0
wcwidth==0.1.7
webencodings==0.5.1
Werkzeug==0.12.2
widgetsnextension==2.0.0
xdiagnose==3.8.4
xkit==0.0.0
XlsxWriter==0.7.3
yarg==0.1.9

Architecture Diagram for Face Mask Detector:



Implementation and Description of Project:

Linear Regression: Simple linear regression statistical method allows us to summarize and study the relationship between two continuous quantitative variables. □ One variable, denoted x , is regarded as the predictor, explanatory, or independent variable. □ The other variable, denoted y , is regarded as the response, outcome, or dependent variable.

Multiple Regression: Multiple regression analysis is used to check whether there is a statistically noteworthy association the middle of sets of variables. It's used to discover patterns in the Numerous relapse Investigation will be very nearly the same Likewise basic straight relapse. The main distinction the middle of straightforward straight relapse Also numerous relapse is in the number for predictors ("x" variables) utilized within those relapse.

The cost Function: Thus let's say, you expanded the size of a specific shop, the place you predicted that those deals might a chance to be higher. Be that in spite of expanding those size, those bargains in that shop didn't expand that a great deal. Thereabouts those expense connected Previously, expanding those span of the shop, provided for you negative outcomes. So, we necessity on minimize these cost. So we present an expense function, which is fundamentally used to characterize and measure those slip of the model.

Gradient Boosting algorithm: Gradient boosting is a machine Taking in strategy to relapse Also arrangement problems, that

produces a prediction model in the structure of an group from claiming powerless prediction models. The exactness of a predictive model might be helped to two ways: . Possibly by grasping characteristic building alternately. Toward applying boosting calculations straight far. There are a significant number boosting calculations in.

□ Gradient Boosting □ XGBoost □ AdaBoost □ Gentle Boost etc. Each boosting algorithm need its own underlying math. Also, a slight variety may be watched same time applying them.

Required tools and Library used for implementation:

- **Numpy**: Used for storing and manipulating high dimensional arrays.
- **Matplotlib**: Used for plotting.
- **OpenCV**: Used for manipulating Images and Video Stream
- **Tensorflow&keras**: Used for designing and training the Face Mask Classifier model
- **Face Detector**: Used for detecting faces with Dual Shot Face Detector.
- **Face Recognition**: Used for detecting facial landmarks.
- **Sklearn**: is a Python module integrating classical machine learning algorithms in the tightly-knit
- **Language Required**: Python 3
- **Software Required**: Jupyter Notebook, kaggle, IDLE, etc.

Merits of House Price Prediction:

House Price prediction, is important to drive Real Estate efficiency. As earlier, House prices were determined by calculating the acquiring and selling price in a locality. Therefore, the House Price prediction model is very essential in filling the information gap and improve Real Estate efficiency.

Prediction house prices are expected to help people who plan to buy a house so they can know the price range in the future, then they can plan their finance well. In addition, house price predictions are also beneficial for property investors to know the trend of housing prices in a certain location.

Data Description:

MSSubClass: Identifies the type of dwelling involved in the sale.

20	1-STORY 1946 & NEWER ALL STYLES
30	1-STORY 1945 & OLDER
40	1-STORY W/FINISHED ATTIC ALL AGES
45	1-1/2 STORY - UNFINISHED ALL AGES
50	1-1/2 STORY FINISHED ALL AGES
60	2-STORY 1946 & NEWER
70	2-STORY 1945 & OLDER
75	2-1/2 STORY ALL AGES
80	SPLIT OR MULTI-LEVEL
85	SPLIT FOYER
90	DUPLEX - ALL STYLES AND AGES
120	1-STORY PUD (Planned Unit Development) - 1946 & NEWER
150	1-1/2 STORY PUD - ALL AGES
160	2-STORY PUD - 1946 & NEWER
180	PUD - MULTILEVEL - INCL SPLIT LEV/FOYER
190	2 FAMILY CONVERSION - ALL STYLES AND AGES

MSZoning: Identifies the general zoning classification of the sale.

A	Agriculture
C	Commercial
FV	Floating Village Residential
I	Industrial
RH	Residential High Density
RL	Residential Low Density
RP	Residential Low Density Park

RM Residential Medium Density

LotFrontage: Linear feet of street connected to property

LotArea: Lot size in square feet

Street: Type of road access to property

Grvl Gravel

Pave Paved

Alley: Type of alley access to property

Grvl Gravel

Pave Paved

NA No alley access

LotShape: General shape of property

Reg Regular

IR1 Slightly irregular

IR2 Moderately Irregular

IR3 Irregular

LandContour: Flatness of the property

Lvl Near Flat/Level

Bnk Banked - Quick and significant rise from street grade to building

HLS Hillside - Significant slope from side to side

Low Depression

Utilities: Type of utilities available

AllPub	All public Utilities (E,G,W,& S)
NoSewr	Electricity, Gas, and Water (Septic Tank)
NoSeWa	Electricity and Gas Only
ELO	Electricity only

LotConfig: Lot configuration

Inside Inside lot

Corner	Corner lot
CulDSac	Cul-de-sac
FR2	Frontage on 2 sides of property
FR3	Frontage on 3 sides of property

LandSlope: Slope of property

Gtl	Gentle slope
Mod	Moderate Slope
Sev	Severe Slope

Neighborhood: Physical locations within Ames city limits

Blmngtn	Bloomington Heights
Blueste	Bluestem
BrDale	Briardale
BrkSide	Brookside
ClearCr	Clear Creek
CollgCr	College Creek
Crawfor	Crawford
Edwards	Edwards
Gilbert	Gilbert

IDOTRR	Iowa DOT and Rail Road
MeadowV	Meadow Village
Mitchel	Mitchell
Names	North Ames
NoRidge	Northridge
NPkVill	Northpark Villa
NridgHt	Northridge Heights
NWAmes	Northwest Ames
OldTown	Old Town
SWISU	South & West of Iowa State University
Sawyer	Sawyer
SawyerW	Sawyer West
Somerst	Somerset
StoneBr	Stone Brook
Timber	Timberland
Veenker	Veenker

Condition1: Proximity to various conditions

Artery	Adjacent to arterial street
Feedr	Adjacent to feeder street
Norm	Normal
RRNn	Within 200' of North-South Railroad
RRAn	Adjacent to North-South Railroad
PosN	Near positive off-site feature--park, greenbelt, etc.
PosA	Adjacent to postive off-site feature
RRNe	Within 200' of East-West Railroad
RR Ae	Adjacent to East-West Railroad

Condition2: Proximity to various conditions (if more than one is present)

Artery Adjacent to arterial street
Feedr Adjacent to feeder street
Norm Normal
RRNn Within 200' of North-South Railroad
RRAn Adjacent to North-South Railroad
PosN Near positive off-site feature--park, greenbelt, etc.
PosA Adjacent to positive off-site feature
RRNe Within 200' of East-West Railroad
RR Ae Adjacent to East-West Railroad

BldgType: Type of dwelling

1Fam Single-family Detached
2FmCon Two-family Conversion; originally built as one-family dwelling
Duplx Duplex
TwnhsE Townhouse End Unit
TwnhsI Townhouse Inside Unit

HouseStyle: Style of dwelling

1Story One story
1.5Fin One and one-half story: 2nd level finished
1.5Unf One and one-half story: 2nd level unfinished
2Story Two story
2.5Fin Two and one-half story: 2nd level finished
2.5Unf Two and one-half story: 2nd level unfinished
SFoyer Split Foyer
SLvl Split Level

OverallQual: Rates the overall material and finish of the house

- 10 Very Excellent
- 9 Excellent
- 8 Very Good
- 7 Good
- 6 Above Average
- 5 Average
- 4 Below Average
- 3 Fair
- 2 Poor
- 1 Very Poor

OverallCond: Rates the overall condition of the house

- 10 Very Excellent
- 9 Excellent
- 8 Very Good
- 7 Good
- 6 Above Average
- 5 Average
- 4 Below Average
- 3 Fair
- 2 Poor
- 1 Very Poor

YearBuilt: Original construction date

YearRemodAdd: Remodel date (same as construction date if no remodeling or additions)

RoofStyle: Type of roof

Flat Flat

Gable Gable
Gambrel Gabrel (Barn)
Hip Hip
Mansard Mansard
Shed Shed

RoofMatl: Roof material

ClyTile Clay or Tile
CompShg Standard (Composite) Shingle
Membran Membrane
Metal Metal
Roll Roll
Tar&Grv Gravel & Tar
WdShake Wood Shakes
WdShngl Wood Shingles

Exterior1st: Exterior covering on house

AsbShng Asbestos Shingles
AsphShn Asphalt Shingles
BrkComm Brick Common
BrkFace Brick Face
CBlock Cinder Block
CemntBd Cement Board
HdBoard Hard Board
ImStucc Imitation Stucco
MetalSd Metal Siding
Other Other
Plywood Plywood
PreCast PreCast
Stone Stone

Stucco	Stucco
VinylSd	Vinyl Siding
Wd Sdng	Wood Siding
WdShing	Wood Shingles

Exterior2nd: Exterior covering on house (if more than one material)

AsbShng	Asbestos Shingles
AsphShn	Asphalt Shingles
BrkComm	Brick Common
BrkFace	Brick Face
CBlock	Cinder Block
CemntBd	Cement Board
HdBoard	Hard Board
ImStucc	Imitation Stucco
MetalSd	Metal Siding
Other	Other
Plywood	Plywood
PreCast	PreCast
Stone	Stone
Stucco	Stucco
VinylSd	Vinyl Siding
Wd Sdng	Wood Siding
WdShing	Wood Shingles

MasVnrType: Masonry veneer type

BrkCmn	Brick Common
BrkFace	Brick Face
CBlock	Cinder Block
None	None
Stone	Stone

MasVnrArea: Masonry veneer area in square feet

ExterQual: Evaluates the quality of the material on the exterior

Ex	Excellent
Gd	Good
TA	Average/Typical
Fa	Fair
Po	Poor

ExterCond: Evaluates the present condition of the material on the exterior

Ex	Excellent
Gd	Good
TA	Average/Typical
Fa	Fair
Po	Poor

Foundation: Type of foundation

BrkTil	Brick & Tile
CBlock	Cinder Block
PConc	Poured Contrete
Slab	Slab
Stone	Stone
Wood	Wood

BsmtQual: Evaluates the height of the basement

Ex	Excellent (100+ inches)
----	-------------------------

Gd Good (90-99 inches)
TA Typical (80-89 inches)
Fa Fair (70-79 inches)
Po Poor (<70 inches)
NA No Basement

BsmtCond: Evaluates the general condition of the basement

Ex Excellent
Gd Good
TA Typical - slight dampness allowed
Fa Fair - dampness or some cracking or settling
Po Poor - Severe cracking, settling, or wetness
NA No Basement

BsmtExposure: Refers to walkout or garden level walls

Gd Good Exposure
Av Average Exposure (split levels or foyers typically score average or above)
Mn Minimum Exposure
No No Exposure
NA No Basement

BsmtFinType1: Rating of basement finished area

GLQ Good Living Quarters
ALQ Average Living Quarters
BLQ Below Average Living Quarters
Rec Average Rec Room
LwQ Low Quality
Unf Unfinished

NA No Basement

BsmtFinSF1: Type 1 finished square feet

BsmtFinType2: Rating of basement finished area (if multiple types)

GLQ Good Living Quarters

ALQ Average Living Quarters

BLQ Below Average Living Quarters

Rec Average Rec Room

LwQ Low Quality

Unf Unfinished

NA No Basement

BsmtFinSF2: Type 2 finished square feet

BsmtUnfSF: Unfinished square feet of basement area

TotalBsmtSF: Total square feet of basement area

Heating: Type of heating

Floor Floor Furnace

GasA Gas forced warm air furnace

GasW Gas hot water or steam heat

Grav Gravity furnace

OthW Hot water or steam heat other than gas

Wall Wall furnace

HeatingQC: Heating quality and condition

Ex Excellent

Gd Good
TA Average/Typical
Fa Fair
Po Poor

CentralAir: Central air conditioning

N No
Y Yes

Electrical: Electrical system

SBrkr Standard Circuit Breakers & Romex

FuseA Fuse Box over 60 AMP and all Romex wiring
(Average)

FuseF 60 AMP Fuse Box and mostly Romex wiring (Fair)

FuseP 60 AMP Fuse Box and mostly knob & tube wiring (poor)

Mix Mixed

1stFlrSF: First Floor square feet

2ndFlrSF: Second floor square feet

LowQualFinSF: Low quality finished square feet (all floors)

GrLivArea: Above grade (ground) living area square feet

BsmtFullBath: Basement full bathrooms

BsmtHalfBath: Basement half bathrooms

FullBath: Full bathrooms above grade

HalfBath: Half baths above grade

Bedroom: Bedrooms above grade (does NOT include basement bedrooms)

Kitchen: Kitchens above grade

KitchenQual: Kitchen quality

Ex Excellent

Gd Good

TA Typical/Average

Fa Fair

Po Poor

TotRmsAbvGrd: Total rooms above grade (does not include bathrooms)

Functional: Home functionality (Assume typical unless deductions are warranted)

Typ Typical Functionality

Min1 Minor Deductions 1

Min2 Minor Deductions 2

Mod Moderate Deductions

Maj1 Major Deductions 1

Maj2 Major Deductions 2

Sev Severely Damaged

Sal Salvage only

Fireplaces: Number of fireplaces

FireplaceQu: Fireplace quality

Ex Excellent - Exceptional Masonry Fireplace

Gd Good - Masonry Fireplace in main level

TA Average - Prefabricated Fireplace in main living area or
Masonry Fireplace in basement

Fa Fair - Prefabricated Fireplace in basement

Po Poor - Ben Franklin Stove

NA No Fireplace

GarageType: Garage location

2Types More than one type of garage

Attchd Attached to home

Basment Basement Garage

BuiltIn Built-In (Garage part of house - typically has room
above garage)

CarPort Car Port

Detchd Detached from home

NA No Garage

GarageYrBlt: Year garage was built

GarageFinish: Interior finish of the garage

Fin Finished

RFn Rough Finished

Unf Unfinished

NA No Garage

GarageCars: Size of garage in car capacity

GarageArea: Size of garage in square feet

GarageQual: Garage quality

Ex	Excellent
Gd	Good
TA	Typical/Average
Fa	Fair
Po	Poor
NA	No Garage

GarageCond: Garage condition

Ex	Excellent
Gd	Good
TA	Typical/Average
Fa	Fair
Po	Poor
NA	No Garage

PavedDrive: Paved driveway

Y	Paved
P	Partial Pavement
N	Dirt/Gravel

WoodDeckSF: Wood deck area in square feet

OpenPorchSF: Open porch area in square feet

EnclosedPorch: Enclosed porch area in square feet

3SsnPorch: Three season porch area in square feet

ScreenPorch: Screen porch area in square feet

PoolArea: Pool area in square feet

PoolQC: Pool quality

Ex Excellent

Gd Good

TA Average/Typical

Fa Fair

NA No Pool

Fence: Fence quality

GdPrv Good Privacy

MnPrv Minimum Privacy

GdWo Good Wood

MnWw Minimum Wood/Wire

NA No Fence

MiscFeature: Miscellaneous feature not covered in other categories

Elev Elevator

Gar2 2nd Garage (if not described in garage section)

Othr Other

Shed Shed (over 100 SF)

TenC Tennis Court

NA None

MiscVal: \$Value of miscellaneous feature

MoSold: Month Sold (MM)

YrSold: Year Sold (YYYY)

SaleType: Type of sale

WD Warranty Deed - Conventional

CWD Warranty Deed - Cash

VWD Warranty Deed - VA Loan

New Home just constructed and sold

COD Court Officer Deed/Estate

Con Contract 15% Down payment regular terms

ConLw Contract Low Down payment and low interest

ConLI Contract Low Interest

ConLD Contract Low Down

Oth Other

SaleCondition: Condition of sale

Normal Normal Sale

Abnorml Abnormal Sale - trade, foreclosure, short sale

AdjLand Adjoining Land Purchase

Alloca Allocation - two linked properties with separate deeds,
typically condo with a garage unit

Family Sale between family members

Partial Home was not completed when last assessed
(associated with New Homes)

Code

```
In [1]: import pandas as pd

In [2]: housing = pd.read_csv("data.csv")

In [3]: housing.head()
Out[3]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98	24.0
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14	21.6
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03	34.7
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33	36.2

```
In [4]: housing.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 506 entries, 0 to 505
Data columns (total 14 columns):
 # Column Non-Null Count Dtype
-----

```

```
In [7]: %matplotlib inline

In [8]: ## For plotting histogram
# import matplotlib.pyplot as plt
# housing.hist(bins=50, figsize=(20, 15))

Train-Test Splitting

In [9]: # For Learning purpose
import numpy as np
def split_train_test(data, test_ratio):
```

Browser tabs: Fwd: 2021-2022 In... | 2021-22 Final Viva... | Inbox (1,719) - ayu... | (10) WhatsApp... | Home Page - Selec... | House Price Predict... | +

Address bar: http://localhost:8888/notebooks/House%20Price%20Prediction%20using%20Supervised%20learning%20.ipynb

Jupyter interface: House Price Prediction using Supervised learning Last Checkpoint: 01/01/2021 (autosaved) | Python 3

Looking for Correlations

```
In [19]: corr_matrix = housing.corr()
corr_matrix['MEDV'].sort_values(ascending=False)

Out[19]: MEDV      1.000000
RM        0.680857
B         0.361761
ZN        0.339741
DIS       0.240451
CHAS      0.205066
AGE       -0.364596
RAD       -0.374693
CRIM      -0.393715
NOX       -0.422873
TAX       -0.456657
INDUS     -0.473516
PTRATIO   -0.493534
LSTAT     -0.740494
Name: MEDV, dtype: float64

In [20]: # from pandas.plotting import scatter_matrix
# attributes = ["MEDV", "RM", "ZN", "LSTAT"]
# scatter_matrix(housing[attributes], figsize = (12,8))

In [21]: housing.plot(kind="scatter", x="RM", y="MEDV", alpha=0.8)
```

Browser tabs: Fwd: 2021-2022 In... | 2021-22 Final Viva... | Inbox (1,719) - ayu... | (10) WhatsApp... | Home Page - Selec... | House Price Predict... | +

Address bar: http://localhost:8888/notebooks/House%20Price%20Prediction%20using%20Supervised%20learning%20.ipynb

Jupyter interface: House Price Prediction using Supervised learning Last Checkpoint: 01/01/2021 (autosaved) | Python 3

```
train_set, test_set = train_test_split(housing, test_size=0.2, random_state=42)
print(f"Rows in train set: {len(train_set)}\nRows in test set: {len(test_set)}\n")

Rows in train set: 404
Rows in test set: 102

In [13]: from sklearn.model_selection import StratifiedShuffleSplit
split = StratifiedShuffleSplit(n_splits=1, test_size=0.2, random_state=42)
for train_index, test_index in split.split(housing, housing['CHAS']):
    strat_train_set = housing.loc[train_index]
    strat_test_set = housing.loc[test_index]

In [14]: strat_test_set['CHAS'].value_counts()

Out[14]: 0    95
         1     7
         Name: CHAS, dtype: int64

In [15]: strat_train_set['CHAS'].value_counts()

Out[15]: 0    376
         1     28
         Name: CHAS, dtype: int64

In [16]: # 95/7

In [17]: # 376/28
```

Trying out Attribute combinations

```
In [22]: housing["TAXRM"] = housing['TAX']/housing['RM']
```

```
In [23]: housing.head()
```

Out[23]:

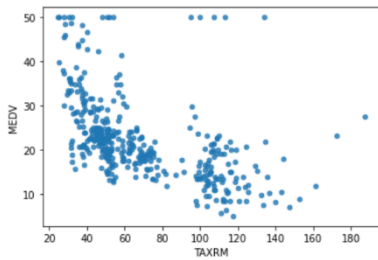
	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	MEDV	TAXRM
254	0.04819	80.0	3.64	0	0.392	6.108	32.0	9.2203	1	315	16.4	392.89	6.57	21.9	51.571709
348	0.01501	80.0	2.01	0	0.435	6.635	29.7	8.3440	4	280	17.0	390.94	5.99	24.5	42.200452
476	4.87141	0.0	18.10	0	0.614	6.484	93.6	2.3053	24	666	20.2	396.21	18.68	16.7	102.714374
321	0.18159	0.0	7.38	0	0.493	6.376	54.3	4.5404	5	287	19.6	396.90	6.87	23.1	45.012547
326	0.30347	0.0	7.38	0	0.493	6.312	28.9	5.4159	5	287	19.6	396.90	6.15	23.0	45.468948

```
In [24]: corr_matrix = housing.corr()  
corr_matrix['MEDV'].sort_values(ascending=False)
```

Out[24]:

MEDV	1.000000
RM	0.680857
B	0.361761
ZN	0.339741
DIS	0.248451
CHAS	0.205066
AGE	-0.364596
TAX	0.374563

```
Out[25]: <AxesSubplot:xlabel='TAXRM', ylabel='MEDV'>
```



```
In [26]: housing = strat_train_set.drop("MEDV", axis=1)  
housing_labels = strat_train_set["MEDV"].copy()
```

Missing Attributes

```
In [27]: # To take care of missing attributes, you have three options:  
# 1. Get rid of the missing data points  
# 2. Get rid of the whole attribute
```

Browser tabs: Fwd: 2021-2022 In... | 2021-22 Final Viva... | Inbox (1,719) - ayu... | (10) WhatsApp... | Home Page - Selec... | House Price Predict... | +

Address bar: http://localhost:8888/notebooks/House%20Price%20Prediction%20using%20Supervised%20learning%20.ipynb

Jupyter interface: House Price Prediction using Supervised learning Last Checkpoint: 01/01/2021 (autosaved) | Python 3

```

In [30]: median = housing["RM"].median() # Compute median for Option 3

In [31]: housing["RM"].fillna(median) # Option 3
# Note that the original housing dataframe will remain unchanged

Out[31]: 254    6.108
348    6.635
476    6.484
321    6.376
326    6.312
...
155    6.152
423    6.103
98     7.820
455    6.525
216    5.888
Name: RM, Length: 404, dtype: float64

In [32]: housing.shape
Out[32]: (404, 13)

In [33]: housing.describe() # before we started filling missing attributes
Out[33]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	L
--	------	----	-------	------	-----	----	-----	-----	-----	-----	---------	---	---

Browser tabs: Fwd: 2021-2022 In... | 2021-22 Final Viva... | Inbox (1,719) - ayu... | (10) WhatsApp... | Home Page - Selec... | House Price Predict... | +

Address bar: http://localhost:8888/notebooks/House%20Price%20Prediction%20using%20Supervised%20learning%20.ipynb

Jupyter interface: House Price Prediction using Supervised learning Last Checkpoint: 01/01/2021 (autosaved) | Python 3

```

In [34]: from sklearn.impute import SimpleImputer
imputer = SimpleImputer(strategy="median")
imputer.fit(housing)

Out[34]: SimpleImputer(strategy='median')

In [35]: imputer.statistics_
Out[35]: array([2.86735e-01, 0.00000e+00, 9.90000e+00, 0.00000e+00, 5.38000e-01,
6.20900e+00, 7.82000e+01, 3.12220e+00, 5.00000e+00, 3.37000e+02,
1.90000e+01, 3.90955e+02, 1.15700e+01])

In [36]: X = imputer.transform(housing)

In [37]: housing_tr = pd.DataFrame(X, columns=housing.columns)

In [38]: housing_tr.describe()
Out[38]:
```

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	L
count	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000	404.000000
mean	3.602814	10.836634	11.344950	0.069307	0.558064	6.278609	69.039851	3.746210	9.735149	412.341584	18.473267	353.392822	12.75
std	8.099383	22.150636	6.877817	0.254290	0.116875	0.712366	28.258248	2.099057	8.731259	168.672623	2.129243	96.069235	7.25
min	0.006320	0.000000	0.740000	0.000000	0.389000	3.561000	2.900000	1.129600	1.000000	187.000000	13.000000	0.320000	1.75

Creating a Pipeline

```
In [39]: from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler
my_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy="median")),
    # ..... add as many as you want in your pipeline
    ('std_scaler', StandardScaler()),
])
```

```
In [40]: housing_num_tr = my_pipeline.fit_transform(housing)
```

```
In [41]: housing_num_tr.shape
```

Out[41]: (404, 13)

Selecting a desired model for Dragon Real Estates

```
In [42]: from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
# model = LinearRegression()
# model = DecisionTreeRegressor()
model = RandomForestRegressor()
```

Evaluating the model

```
In [48]: from sklearn.metrics import mean_squared_error
housing_predictions = model.predict(housing_num_tr)
mse = mean_squared_error(housing_labels, housing_predictions)
rmse = np.sqrt(mse)
```

```
In [49]: rmse
```

Out[49]: 1.2098819187944505

Using better evaluation technique - Cross Validation

```
In [50]: # 1 2 3 4 5 6 7 8 9 10
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, housing_num_tr, housing_labels, scoring="neg_mean_squared_error", cv=10)
rmse_scores = np.sqrt(-scores)
```

```
In [51]: rmse_scores
```

Out[51]: array([2.82810603, 2.99020411, 4.44317235, 2.70275527, 3.47379099,
2.69137442, 4.75478438, 3.37070364, 3.21647979, 3.35649503])

```
In [52]: def print_scores(scores):
print("Scores:", scores)
```

```
X_test_prepared = my_pipeline.transform(X_test)
final_predictions = model.predict(X_test_prepared)
final_rmse = mean_squared_error(Y_test, final_predictions)
final_rmse = np.sqrt(final_rmse)
# print(final_predictions, list(Y_test))

In [56]: final_rmse
Out[56]: 2.9650820790146972

In [57]: prepared_data[0]
Out[57]: array([-0.43942006,  3.12628155, -1.12165014, -0.27288841, -1.42262747,
               -0.23979304, -1.31238772,  2.61111401, -1.0016859 , -0.5778192 ,
               -0.97491834,  0.41164221, -0.86091034])

Using the model

In [58]: from joblib import dump, load
import numpy as np
model = load('Dragon.joblib')
features = np.array([[-5.43942006, 4.12628155, -1.6165014, -0.67288841, -11.42262747,
                    -15.44443979304, -49.31238772, 712.61111401, -262.0016879 , -0.5778192 ,
                    -0.974918374, 01.41164221, -66.86091034]])
model.predict(features)

Out[58]: array([24.38])
```

Results

Many iterations of performance tuning were done to find the optimal solution of each model. Random Forest Regression, XGBoost, and LightGBM were intensively tuned by function GridSearchCV provided by scikit-learn [9] to achieve the results listed in Table 2. For Hybrid Regression and Stacking methods, performance tuning was not required since both methods were combinations of the best regressions. Instead, architecture implementation could be considered to further enhance the prediction.

Table 2. Prediction Results

As listed in the Table 2, the best results belong to Random Forest for the training set and Stacked Generalization Regression for the test set.

Since 49 of 58 features of the one-hot-encoded dataset were boolean values, it is reasonable that the Random Forest worked well on this dataset. However, Random Forest was prone to overfitting, which led to a decent performance on unseen data. Both XGBoost and LightGBM were not subject to overfitting, but the accuracy of their predictions was not as good as Random Forest on both training and test data.

Unlike the three traditional machine learning methods, Hybrid Regression and Stacked Generalization Regression were neither tuned nor implemented sophisticatedly but delivered promising results on the training set and test set.

Since Random Forest was proven to be overfitting, Hybrid Regression could be considered as the best model on training set where the RSMLE is 0.14969. Surprisingly, Stacked Generalization Regression did not work well on the training set as Hybrid Regression, but this model did exceptionally on the test set. This is probably because of the two reasons

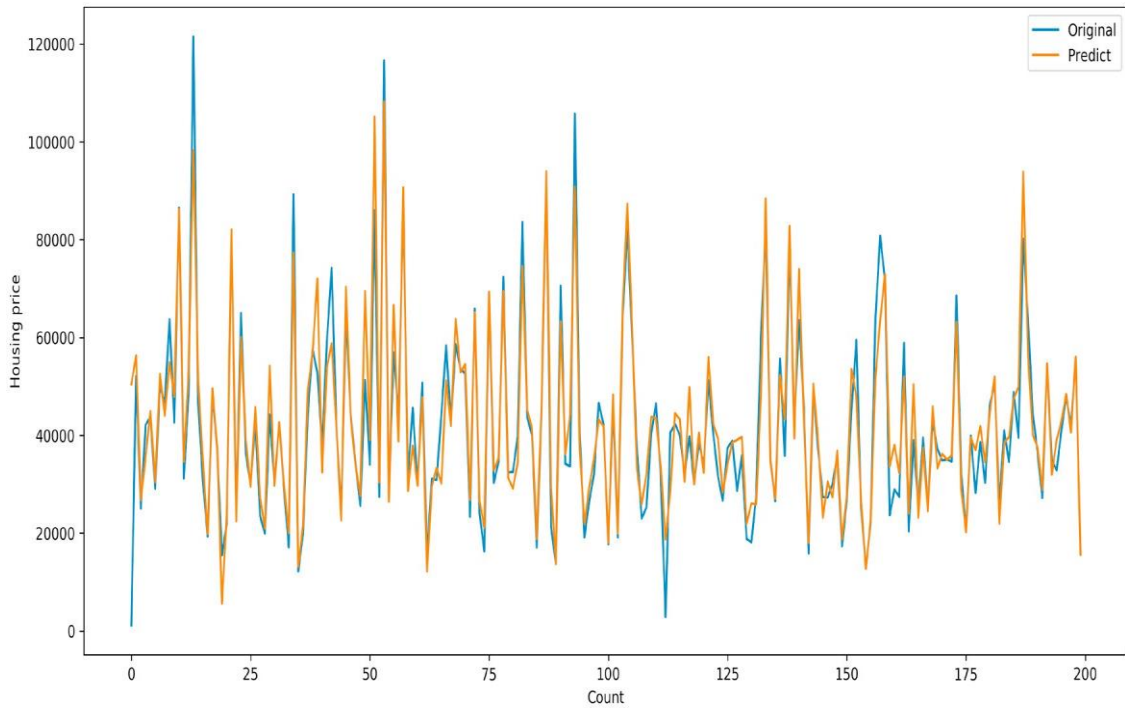


Fig. 12. Comparison of Hybrid Regression's predicted results and original test set

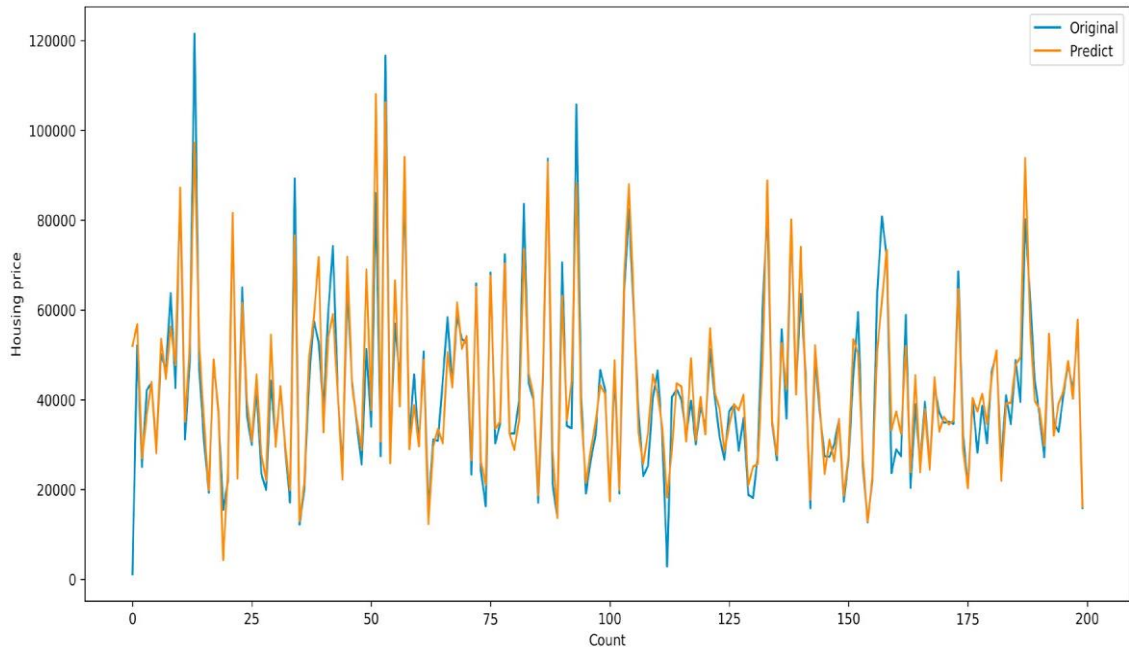


Fig. 13. Comparison of Stacked Generalization Regression's predicted results and original test set

- **K-fold cross-validation:** k-fold cross-validation is a suitable method to find an acceptable bias-variance trade-off. Stacking Regression utilizes this technique to obtain the generalization performance for each component model.
- **Coupling effect of multiple regressions:** different regression methods may support each other. The second stacking level can learn again and predict the housing prices accurately based on the pre-estimated prices from the first stacking level.

Even though Hybrid Regression shares the coupling effect of multiple regressions with Stacked Generalization Regression, Hybrid Regression is less competitive because its learning mechanism is only averaging every prediction. Based on Fig. 12 and Fig. 13 which illustrate the performance of both models on the first 200 observations of the test set, the difference between the two models is not considerably large. On one hand, Hybrid Regression predicts more accurately on outliers (extremely high or low housing prices). On the other hand, Stacked Generalization Regression performs better on common observations (standard housing prices). This proves that Stacked Generalization Regression is slightly better than Hybrid Regression in generalization.

Discussion and Conclusion

This paper investigates different models for housing price prediction. Three different types of Machine Learning methods including Random Forest, XGBoost, and LightGBM and two techniques in machine learning including Hybrid Regression and Stacked Generalization Regression are compared and analyzed for optimal solutions. Even though all of those methods achieved desirable results, different models have their own pros and cons. The Random Forest method has the lowest error on the training set but is prone to be overfitting. Its time complexity is high since the dataset has to be fit multiple times. The XGBoost and LightGBM are decent methods when comparing accuracy, but their time complexities are the best, especially LightGBM. The Hybrid Regression method is simple but performs a lot better than the three previous methods due to the generalization. Finally, the Stacked Generalization Regression method has a complicated architecture, but it is the best choice when accuracy is the top priority. Even though Hybrid Regression and Stacked Generalization Regression deliver satisfactory results, time complexity must be taken into consideration since both of them contain Random Forest, a high time complexity model. Stacked Generalization Regression also has K-fold cross-validation in its mechanism so it has the worst time complexity. Further research about the following topics should be conducted to further investigate these models, especially the combinations of different models:

- The coupling effect of multiple regression models.
- The “re-learn” ability of machine learning models.
- The combination of Machine Learning and Deep Learning methods.
- The driven factors for the good performance of tree-based models.
- The faster ways to fit complex models.

Summary:

We have managed out how to prepare a model that gives users for a novel best approach with take a gander at future lodging value predictions. A few relapse strategies have been investigated Furthermore compared, when arriving during a prediction strategy In light of XG support. Straight former imply works bring been utilized within our model, something like that that future value predictions will have a tendency towards All the more sensible values. We concocted an approach with use similarly as considerably information as time permits for our prediction system, by adopting those ideas from claiming gradient boosting.

References

- [1] House Price Index. Federal Housing Finance Agency. <https://www.fhfa.gov/> (accessed September 1, 2019).
- [2] Fan C, Cui Z, Zhong X. House Prices Prediction with Machine Learning Algorithms. Proceedings of the 2018 10th International Conference on Machine Learning and Computing - ICMLC 2018. doi:10.1145/3195106.3195133.
- [3] Phan TD. Housing Price Prediction Using Machine Learning Algorithms: The Case of Melbourne City, Australia. 2018 International Conference on Machine Learning and Data Engineering (ICMLDE) 2018. doi:10.1109/icmlde.2018.00017.
- [4] Mu J, Wu F, Zhang A. Housing Value Forecasting Based on Machine Learning Methods. Abstract and Applied Analysis 2014;2014:1–7. doi:10.1155/2014/648047.
- [5] Lu S, Li Z, Qin Z, Yang X, Goh RSM. A hybrid regression technique for house prices prediction. 2017 IEEE International Conference on Industrial Engineering and Engineering Management (IEEM) 2017. doi:10.1109/ieem.2017.8289904.
- [6] Chen T, Guestrin C. XGBoost. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD 16 2016. doi:10.1145/2939672.2939785.
- [7] DMLC. xgboost. GitHub. <https://github.com/dmlc/xgboost> (accessed June 1, 2019).
- [8] Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, et al. LightGBM: A Highly Efficient Gradient Boosting Decision Tree. Advances in Neural Information Processing Systems

Proof of Research Paper Submission:

