Phase 5 submission document Project Title: House Price Predictor

Phase 5: Project Documentation & Submission Topic: In this section we will document the complete project and prepare it for submission.

***Topic**: In this section we will document the complete project and prepare it for submission.*

N.Manikandan

AU712521104018

ppg intitution of technology

BE CSE

Introduction

The real estate market is a complex and dynamic environment influenced by various factors such as location, size, amenities, economic conditions, and market trends. Understanding and predicting house prices is of paramount importance for homeowners, real estate professionals, and investors alike. In recent years, advancements in data science and machine learning techniques have opened up new possibilities for more accurate and reliable predictions.

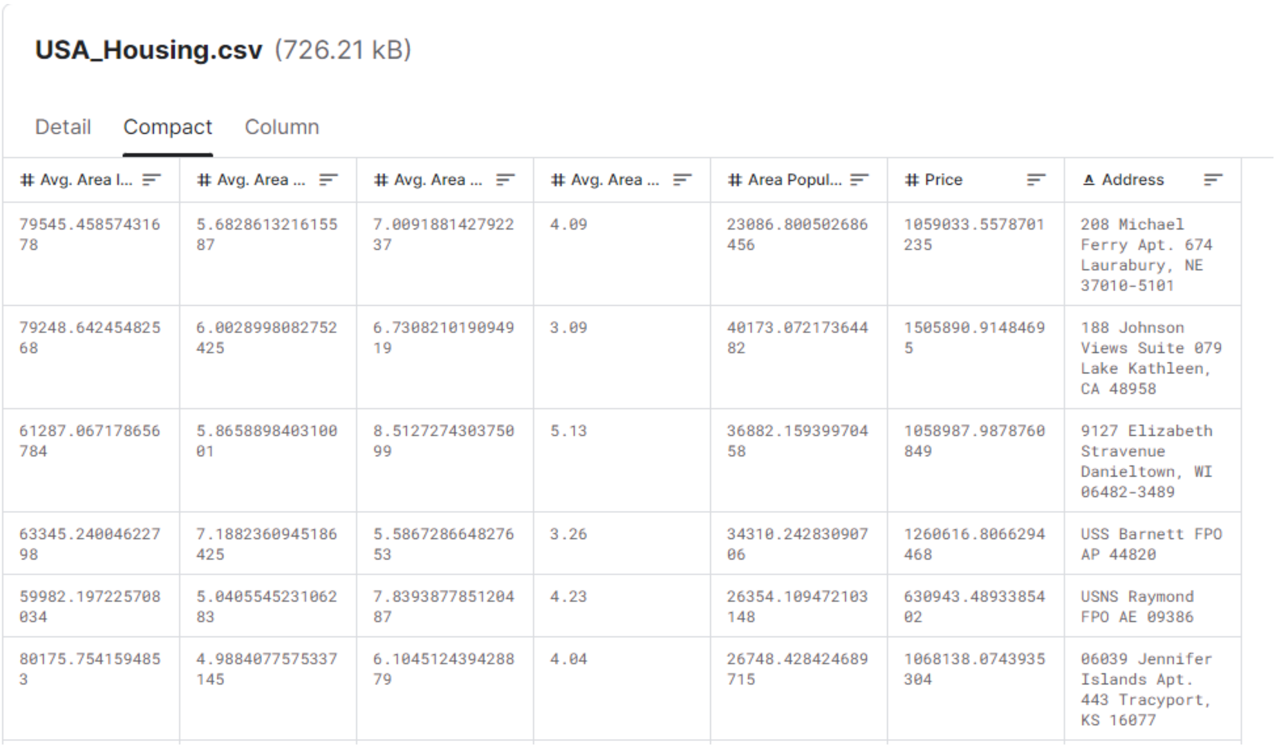
This project aims to delve into the realm of house price prediction using advanced data analytics and machine learning algorithms. By leveraging historical property data and relevant features, we seek to develop a robust model capable of estimating house prices with a high degree of accuracy. Such a model has the potential to assist homeowners in setting competitive listing prices, aid buyers in making informed decisions, and provide valuable insights for real estate professionals and investors.

The dataset for this analysis encompasses a comprehensive set of features, including but not limited to property size, location, number of bedrooms and bathrooms, proximity to amenities, and economic indicators. Through exploratory data analysis (EDA) and feature engineering, we will uncover patterns, correlations, and hidden insights within the data, paving the way for an effective predictive model.

Machine learning algorithms, including regression models, ensemble methods, and neural networks, will be employed to train and evaluate the predictive model. The model's performance will be assessed based on metrics such as mean absolute error, mean squared error, and R-squared to ensure its reliability and generalizability.

In conclusion, this project strives to contribute to the field of real estate analytics by developing a robust house price prediction model. The insights gained from this analysis have the potential to empower stakeholders in the real estate market with data-driven decision-making capabilities, ultimately fostering a more transparent and efficient housing market.

Dataset Link: (https://www.kaggle.com/datasets/vedavyasv/usa-housing) 5000 Rows x 7 Columns



USA_Housing.csv (726.21 kB)

Detail Compact Column

# Avg. Area I...	# Avg. Area ...	# Avg. Area ...	# Avg. Area ...	# Area Popul...	# Price	Address
79545.45857431678	5.682861321615587	7.009188142792237	4.09	23086.800502686456	1059033.5578701235	208 Michael Ferry Apt. 674 Laurabury, NE 37010-5101
79248.64245482568	6.002899808275425	6.730821019094919	3.09	40173.07217364482	1505890.91484695	188 Johnson Views Suite 079 Lake Kathleen, CA 48958
61287.067178656784	5.86588984031001	8.512727430375099	5.13	36882.15939970458	1058987.9878760849	9127 Elizabeth Stravenue Danieltown, WI 06482-3489
63345.24004622798	7.188236094518625	5.586728664827653	3.26	34310.24283090706	1260616.8066294468	USS Barnett FPO AP 44820
59982.197225708034	5.040554523106283	7.839387785120487	4.23	26354.109472103148	630943.4893385402	USNS Raymond FPO AE 09386
80175.7541594853	4.988407757533145	6.104512439428879	4.04	26748.428424689715	1068138.0743935304	06039 Jennifer Islands Apt. 443 Tracyport, KS 16077

Here's a list of tools and software commonly used in theprocess:

1. Programming Language: - Python is the most popular language for machine learning due toits extensive libraries and frameworks. You can use libraries likeNumPy,pandas, scikit-learn, and more.

2. Integrated Development Environment (IDE): - Choose an IDE for coding and running machine learningexperiments. Some popular options include Jupyter Notebook, GoogleColab, or traditional IDEs like PyCharm.

3. Machine Learning Libraries: - You'll need various machine learning libraries, including: - scikit-learn for building and evaluating machine learning models. - TensorFlow or PyTorch for deep learning, if needed. - XGBoost, LightGBM, or CatBoost for gradient boosting models.

4. Data Visualization Tools: - Tools like Matplotlib, Seaborn, or Plotly are essential for dataexploration and visualization.

5. Data Preprocessing Tools: - Libraries like pandas help with data cleaning, manipulation, andpreprocessing. .

6. Data Collection and Storage: - Depending on your data source, you might need web scrapingtools (e.g., BeautifulSoup or Scrapy) or databases (e.g., SQLite, PostgreSQL) for data storage.

7. Version Control: - Version control systems like Git are valuable for trackingchanges in your code and collaborating with others.

8. Notebooks and Documentation: - Tools for documenting your work, such as Jupyter Notebooksor Markdown for creating README files and documentation.

9. Hyperparameter Tuning: - Tools like GridSearchCV or RandomizedSearchCV fromscikit-learn can help with hyperparameter tuning.

10. Web Development Tools (for Deployment): - If you plan to create a web application for model deployment, knowledge of web development tools like Flask or Django for backenddevelopment, and HTML, CSS, and JavaScript for the front-end can beuseful.

11. Cloud Services (for Scalability): - For large-scale applications, cloud platforms like AWS, GoogleCloud, or Azure can provide scalable computing and storage resources.

12.External Data Sources (if applicable): - Depending on your project's scope, you might require tools toaccess external data sources, such as APIs or data scraping tools.

13. Data Annotation and Labeling Tools (if applicable): - For specialized projects, tools for data annotation andlabeling may be necessary, such as Labelbox or Supervisely.

14. Geospatial Tools (for location-based features): - If your dataset includes geospatial data, geospatial librarieslike GeoPandas can be helpful.

1.DESIGN THINKING AND PRESENT IN FORM OF DOCUMENT

Definition: House price prediction involves the development of models or algorithms that can estimate the value of residential properties based on various factors. These factors typically include features such as location, size, number of bedrooms and bathrooms, amenities, and economic indicators. The goal is to create a predictive model that can analyze historical data and learn patterns, enabling it to make accurate predictions about the prices of houses in a given market.

The prediction models can be implemented using machine learning algorithms, statistical methods, or a combination of both. These models aim to provide valuable insights to homeowners, real estate professionals, and investors, allowing them to make informed decisions about buying, selling, or investing in properties.

Design Thinking for House Price Prediction:

Design thinking is a problem-solving approach that emphasizes empathy, ideation, and prototyping to address complex challenges. When applied to house price prediction, the design thinking process can help create user-centric solutions that meet the needs of various stakeholders in the real estate market. Here's how design thinking principles can be applied:

1. **Empathize:**
    - Understand the needs and pain points of key stakeholders, such as homeowners, buyers, real estate agents, and investors.
    - Conduct interviews, surveys, or observations to gather insights into their perspectives on the house pricing process.
2. **Define:**
    - Clearly define the problem and the goals of the house price prediction system.
    - Identify specific features and requirements that are crucial for accurate predictions and user satisfaction.
3. **Ideate:**
    - Brainstorm potential features and data sources that can enhance the accuracy of the prediction model.

- Encourage creative thinking to explore innovative ways to visualize and communicate predicted house prices.

4. **Prototype:**

- Develop prototypes of the predictive model, incorporating the identified features and data sources.
- Create user interfaces or dashboards that make the predictions easily accessible and understandable for different user groups.

5. **Test:**

- Gather feedback from users and stakeholders on the prototype.
- Test the model's predictions against real-world scenarios and refine it based on user input.

6. **Iterate:**

- Iterate on the model and its interface based on the feedback received during testing.
- Continuously refine and improve the model as new data becomes available or as market conditions change.

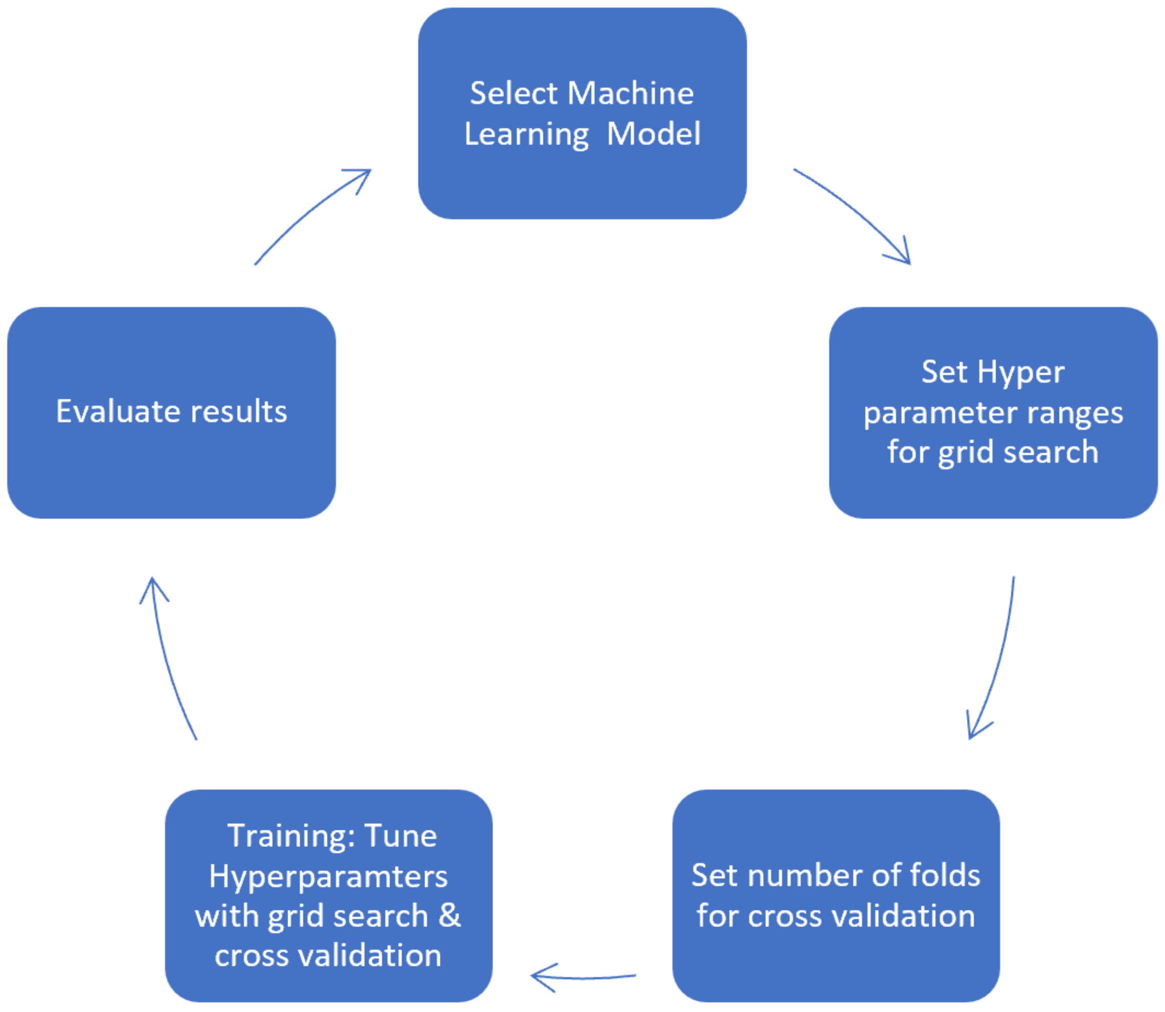
7. **Implement:**

- Implement the final version of the house price prediction model in a real-world setting.
- Ensure seamless integration with existing real estate platforms or tools used by stakeholders.

8. **Monitor and Adapt:**

- Monitor the performance of the model in a live environment.
- Adapt the model as needed to address emerging challenges or changes in user requirements.

By applying design thinking principles, the house price prediction system can be tailored to meet the specific needs of users, ensuring that it is not only accurate but also user-friendly and aligned with the dynamics of the real estate market.


Select Machine Learning Model
Set Hyper parameter ranges for grid search
Set number of folds for cross validation
Training: Tune Hyperparamters with grid search & cross validation
Evaluate results

2. **how to solving the house price prediction**

Solving the house price prediction challenge involves a systematic approach that combines data preprocessing, exploratory data analysis (EDA), feature engineering, and the application of machine learning algorithms. Here's a step-by-step guide on how to approach the problem:

1. **Data Collection and Understanding:**
    - Gather a comprehensive dataset that includes relevant information about the properties, such as size, location, number of bedrooms and bathrooms, amenities, and historical price data.
    - Understand the dataset by exploring its structure, identifying missing values, and gaining insights into the distribution of key variables.
2. **Data Preprocessing:**
    - Handle missing data through imputation or removal.
    - Encode categorical variables and normalize numerical features to ensure consistency in the dataset.
    - Address outliers that might skew the model's predictions.
3. **Exploratory Data Analysis (EDA):**
    - Conduct EDA to understand the relationships between different features and the target variable (house prices).
    - Visualize data distributions, correlations, and trends.
    - Identify potential feature engineering opportunities based on EDA insights.
4. **Feature Engineering:**
    - Create new meaningful features that might enhance the predictive power of the model.
    - Transform variables to make them more suitable for the chosen machine learning algorithms.
    - Consider the inclusion of interaction terms or polynomial features to capture non-linear relationships.
5. **Split the Data:**
    - Divide the dataset into training and testing sets to train the model and evaluate its performance on unseen data.
6. **Model Selection:**
    - Choose appropriate machine learning algorithms for regression, such as linear regression, decision trees, random forests, or gradient boosting.
    - Experiment with different algorithms to find the one that best fits the data and problem at hand.
7. **Model Training:**

- Train the selected models on the training dataset.
- Optimize hyperparameters to improve model performance.

8. **Model Evaluation:**
    - Evaluate the models using metrics such as mean absolute error, mean squared error, and R-squared on the testing dataset.
    - Assess the model's performance and generalization ability.
9. **Fine-Tuning:**
    - Fine-tune the model based on evaluation results.
    - Consider ensemble methods or more complex models for improved performance.
10. **Deployment:**

- Once satisfied with the model's performance, deploy it to predict house prices for new, unseen data.

11. **Monitoring and Updating:**

- Regularly monitor the model's performance in a production environment.
- Update the model as needed to account for changes in the real estate market or dataset characteristics.

By following these steps, you can build a robust house price prediction model that leverages the power of data science and machine learning. Keep in mind that the success of the model depends on the quality of the data, feature engineering, and the appropriateness of the chosen algorithms.

2.An Innovative Method to Predict house Prices

INTRODUCTION

Real estate, like any other commodity, has value because of its practicality as well as its monetary worth. Value is the foundation of all real estate transactions. In general, real estate pricing can be viewed as the market's reflection of a property's true worth. There are, however, subtleties to keep in mind. When compared to the pricing of other commodities, real estate is priced using a rather unique method. Real estate pricing, the economics of determining the worth of land and buildings, is a particularly intricate field. In addition, as was previously said, the environment is merely one of many general and specialized aspects that determine the cost of real estate. There are other factors at play, such as supply-side policy, trade volume, and interest rates. Studies that examine the impacts of certain environmental factors on real estate value examine the significance of regional ecosystems. There are several factors to think about including proximity to setting, public facilities, environmental friendliness, population density, household characteristics, accessibility via public transportation, kind of residence, size, purpose, number of rooms, floor level and age of building. The real estate market frequently uses hedonic pricing. Appraisals of both real estate and personal property rely significantly on market observation in order to identify any abnormal price movements. In this respect, accurate forecasts and representations of future home prices are crucial. It is crucial when geographically restricted property data must be used to monitor investment risks. Under these circumstances, the ability to predict home price changes in unsampled areas is essential. The use of hedonic models is widespread. According to this school of thought, a piece of property is worth the sum of money that may be exchanged for it due to the utility it offers. Hedonic regression is used to calculate the effect of home and community characteristics on selling prices (such as square footage and demographics). Housing prices are known to fluctuate geographically due to factors including the permanence of individual homes and the demand for real estate in specific areas. The value of real estate is often used as a stand-in for the health of the economy. Homeowners in various parts of the economy might benefit from real estate price change indices since they provide a better understanding of the worth of their property. Real estate price indices are commonly calculated using hedonic regression or the repeat-sale method. According to the hedonic viewpoint, the observed price change can be decomposed into classes of attributes with varying monetary worth. Property prices can be broken down into their component pieces, each of which reflects the underlying structural and locational pricing, which can then be viewed as the total observed price of a property. Hedonic procedures estimate a real estate price index using data on covariates and dummy variables standing in for important features, and then utilize regression methods to account for various forms of price heterogeneity. Hedonic regressions may be susceptible to model specification bias because of the covariates they use, however, due to data limitations and the researcher's own value judgments. They claimed that inapplicable hedonic variables can skew regression results obtained using the hedonic approach. There is a lack of complexity, clarity, and potentially wide variation in the way hedonic information relates to sales prices. There is no surefire way to quantify the value of the property. Subjective methods, such as relying on one's "gut" or "years of experience," are not sufficient for real estate valuation. This makes use of classical methods in addition to statistical ones. Values calculated using these methods don't always line up with market values. Identifying the criteria used to assign a value to a property and settling on a method that produces objective results is the first step in getting rid of such disparities. A property's value is based on a number of elements, including its location, legal status, construction quality, physical condition, and environmental hazards. These characteristics can vary greatly in both quantity and quality.

II. LITERATURE SURVEY

Some studies that attempted to predict future property values are provided below. Using large data, [1] developed three models random forest, linear regression, and gradient boosting to forecast house sale data in Iowa. The numerical evidence suggests that the predictions made by the gradient boosting model are the most reliable. [2]used evolutionary algorithms to forecast rental costs when four distinct property characteristics and locations were taken into account. The same set of data was used in conjunction with multivariate regression to make forecasts. Numerical analysis shows that genetic algorithms perform better than multivariate regression at making predictions. [3] used complex statistical models based on assumptions about the variance process to forecast US housing market volatility utilizing a logistic smooth transition autoregressive fractionally integrated process. Both the Markov-switching multifractal and the fractionally integrated generalized autoregressive conditional heteroscedastic models have been shown to perform well in simulations. [4] developed a model to forecast the price of South Korean using news articles. Our independent variable was the number of times key terms were used in online news searches. Numerical testing demonstrated that the proposed models outperformed time series methods. The real estate entity consists of the land and structures as well as the owner's rights, duties, and benefits. As a result, real estate valuation is an effort to assign a monetary value to the benefits and drawbacks associated with property ownership [5]. Expertise in appraising and assessing real estate is best left to professionals. The specialists' assessment needs to be grounded in the standard economic principles that control the real estate industry. Concepts like supply and competition, demand, flexibility, variety, and innovation all fall under this umbrella. All of these regulations affect a property's efficiency and usefulness, albeit in different ways. Therefore, it is reasonable to conclude that a property's utility reflects the sum total of the effects of all market forces on its value. Model in [6] uses a database of hedonic qualities and coefficients to predict real estate prices, as well as data from recently closed projects. Because of its high function approximation capabilities, a fuzzy neural network prediction model is shown to be effective in predicting real estate prices. The difficulty of creating real estate databases has been solved, according to researchers at Colombia's Universidad de los Andes [7].The statistical information gathered from the database is then used to analyze the potential impact of threats like land invasions and expropriations on property values. In another publication [8], the authors detail a case study conducted with a representative database of urban real estate sales in a medium-sized city in the South of Spain. Accurate home price predictions, reduced bias across markets, improved use of category data, and increased efficiency thanks to the implementation of Artificial Neural Networks. This method details the steps involved in applying data mining algorithms to real sales data gathered by an assessment office in a major US city [9]. The results are then used to estimate future home prices based on trends in comparable sales. [10]compared three machine learning methods Support Vector Machine (SVM), RF, and gradient boosting machine (GBM) to examine 41,000 Hong Kong real estate transactions. The Chinese real estate market was evaluated by [11]using a back-propagation (BP) neural network. The system concluded that, from a technological aspect, the model's prospective application in pricing real estate was feasible. An algorithm named arrange plunge calculation was developed by[12], and it considerably reduces the effort of computations by limiting the number of highlights while picking the most important ones. Real estate transactions in India are always finalized at the seller's convenience. Buying a home in India can be laden with potential for prejudice due to the lack of a standardized manner of promoting the asking price. However, established media outlets can manipulate news to advance their own agendas[13]. Given the regularity with which home values rise, a method for predicting future price rises is essential [14].A home's expected appreciation or depreciation can help developers and buyers decide when is the optimum time to buy. [15]have studied the unpredictable and nonlinear trends in real estate price fluctuations. The neural network approach, which can be used without a mathematical model, has been implemented. Using back propagation neural networks (BPNs) and distributed radial basis functions (RBFs), a nonlinear model for the projected home price range was built. Finally, the authors here [16] have undertaken similar research comparing several Machine Learning calculations for predicting land value patterns. These calculations include Linear Regression with gradient descent, K nearest neighbor, and Random Forest. The purpose of this proposed system is to determine which of several possible methods for performing such machine learning computations is the most practical.

The real estate market in Santiago, Chile is evaluated, utilizing real data collected by machine learning methods[17]. The method's goal is to evaluate the predictive efficacy of Ordinary Least Squares Regression in comparison to other approaches, such as Random Forests, Neural Networks, and Support Vector Machines. Using search engine query data and machine learning methods, [18]accurately forecasted the jobless rate. Real estate price forecasting using a TEI@I integrated approach was created by [19]after they identified gaps in the existing research. In preparation for a future where the Internet and other Information technologies are pervasive, more and more people are turning to the Internet as a source of real estate market data. Developed a system for estimating property values using SVR. The real estate market is a major contributor to national economies worldwide. Buyers, sellers, and economists can all benefit from taking note of market activity and making informed price forecasts. Real estate forecasting is a complex and challenging undertaking due to the many direct and indirect components that invariably influence the accuracy of estimates. The price of a home can be affected by a wide variety of quantitative and qualitative factors. Potential factors in the quantitative analysis include macroeconomic variables, business cycles, and property attributes.

III. PROPOSED SYSTEM

The cost of housing has a big impact on people, families, businesses, and governments. Online businesses like Zillow recently created their own proprietary algorithms that offer automatic estimations of home prices without the urgent requirement for experienced appraisers. Our knowledge of the factors that influence home values, however, is very limited. In this system, to integrate data from numerous sources to assess the economic impact of neighborhood attributes like walkability and perceived safety. And also create and make available a system that uses open data to forecast house values in the present without the requirement for previous transactions. A. Preprocessing of Data: 1) Normalization: The proposed methodology is divided into three main sections: normalization, machine learning modeling, and data preprocessing. To get it ready for the modeling stage, the raw CDR data goes through a number of functions. Eleven construction blocks can be used to break up data preprocessing. The input data has been cleaned, and the locations of the home and workplace have been identified. Due to the discrepancies in the parameter values and dynamic range, the normalizing technique is used in this method of. Equation (1) can be used to construct and execute this strategy for converting values measured on several scales to a scale that is ostensibly common among parameters for the ranges from +1 to -1. The minimum and maximum input values can be used to generate the final values, which can range from +1 to -1. The inaccuracies caused by variations in the parameter range could be greatly reduced by using the normalization procedure. $wM = ((\ w - wmin\ wmax - wmin\ ) \times 2) - 1$ (1) Where wM stands for normalized data between +1 and -1. In the dataset, $wmin$ and $wmax$ are the lowest and highest numbers, respectively. B. Feature Selection The process of selecting a subset of input features to use in order to improve the generated model is known variously as feature selection, feature subset selection (FSS), and attribute selection (Attribute Selection). Feature redundancy and feature dependence are common in practical machine learning applications. Eliminating unnecessary or redundant features from the data may help it perform better. The goal of speeding up the process can be attained as well. On the other hand, choosing the features that are truly relevant might simplify the model and make the researchers' understanding of the data generating process. 1) PCA Feature Selection A statistical analytic technique to identify a thing's fundamental contradiction is called Principle Component Analysis (PCA). This approach's fundamental goal is to simplify difficult situations and uncover the nature of things by addressing their primary causes. For this reason, principal component analysis (PCA) is frequently employed. Principal component analysis (PCA) is a statistical method for representing high-dimensional data with fewer variables. When broken down into its component drawings, some of the elements that make up a multidimensional vector are difficult to tell apart. Finding and eliminating components that have undergone major modifications or dimensions that have experienced enormous alterations might speed up the computing process. 2) SF – Feature Selection The Sequential Forward Selection (SFS) algorithm is an example of a heuristic search method. Starting with an empty feature subset W, this method progressively fills it with features until the Criterion function ($I(W)$) is maximized. In practice, it is a simple greedy algorithm that selects a feature that will, on average, always yield the best result for the evaluation function. The problem is that once a feature is added to subnet W, it cannot be deleted, which could generate nesting concerns. If one quality, A , is wholly reliant on the other two qualities, A and D, then it is clear that A is irrelevant because A and D are the dominant features. If the Sequential Forward Selection method picks feature B first, then adds features A and D, the resulting subset will be redundant because feature A could not be eliminated [24] . The Sequential Backward Feature Selection (SBS) technique, a different sequential feature selection methodology, uses the opposite mechanism. Starting with the whole set $W = Z$, one characteristic is removed each time to ensure that the evaluation function is optimal. The layering problem is another flaw in this approach. The deleted functionality cannot be added back, and it requires more calculation than SFS. As a result, the selection in this part is made using the Sequential Forward Selection. SFS's selection procedure resembles a search procedure. The operation begins with an empty set and adds features to it one at a time until the evaluation function is optimized, at which point either all features have been added to the subset or the set is full. C. Training the Model: 1) CBAM : Here, they'll employ a two-dimensional method to distribute the emphasis, one that takes into account both the channel and the physical location. The primary components of the convolutional block attention module (CBAM) are the channel attention mechanism and the spatial attention mechanism, respectively. To simplify, the input features IEEE 1249 that lead to the output features are pooled, their weights are distributed, and any overlaps are merged using the channel attention approach. Once the sigmoid function is used, the answer becomes clear. The fundamental goal of the channel attention mechanism is to determine whether input image's content is more crucial. Every pixel on the image is processed using average pooling to obtain feedback. Regarding the maximum pooling, only the area in the image with the highest response will receive feedback. Consequently, the following is how the channel attention process might be expressed:

$Nt(G) = X(MLP(Avg\ Pool(G))) + MLP(Max\ pool(G))$ (2) $= X(X1(G0(avg\ d)) + X1(X0(Gmax\ d)))$ (3)

Where avg denotes average pooling, Max denotes maximum pooling, and MLP denotes weight sharing. Sigmoid function is donated by the operator. Similar to this, the mechanism for spatial attention can be thought of as channel compression. Maximum pooling's purpose is to take the most value possible from the channel. The height times the width equals the number of extractions. The mean value of the channel is extracted by average pooling, with the total number of samples being proportional to the square of the channel's height and width. Then, the 2- channel characteristics are included with the 1-channel characteristics. Here is one approach to describe the brain's spatial attention mechanism:

$Nt(G) = X(G^{7\times7}[(Avg\ Pool(G)); Max\ Pool(G)]) = X(G^{7\times7}([avg\ t; Gmax\ t]))$ (4)

Let's pretend they have a 7 × 7 -size convolution kernel, where Avg and max stand for average and maximum pooling. Here, the convolutional outputs of each block are subjected to the convolutional block attention module (CBAM). The CBAM has been included to the ResNet50 system. 2) GRU Traditional recurrent neural networks (RNNs) are particularly good at handling sequential input, but they run the danger of losing information if they become too large and can't connect to every node. Therefore, it is unable to address the issue of long-distance reliance, and performance may suffer greatly. To choose the GRU network in this method, a version of the LSTM, due to this flaw in the conventional RNN network. Compared to the LSTM neural network, it performs better and has a simpler topology. In the GRU network, Us and Ys stand in for the update gate and reset gate, respectively. The update gate represents the degree of change, or the percentage of state information from the previous moment that can enter the current state, whereas the reset gate determines how much data will be written into the current candidate state. Given the input Ws, the reset gate produces the following result:

$us = X(uf\ [Es-1, ws])$ (5) For the hidden layer, it contains the output state, $Es-1$, the weight on the reset gate, Xu, and Y the sigmoid function,. The output of the update gate is as follows: $s =$

Xyf $[Es-1 , ws])$ (6) In this equation, represents the sigmoid function, $Es-1$ the output state of the hidden layer at the previous instant, and Xy the weight on the update gate. The output result for the candidate state is as follows.

$\tilde{E}s = \tan e(X\tilde{E}f[usfEs-1 , ws])$ (7)

The activation function tane is applied to the output state $Es-1$ of the hidden layer, while $X\tilde{E}$ represents the weight placed on the candidate state. The following is the output state of the hidden layer: $Es = (1 - ys). Es-1 + ys . \tilde{E}s$ (8) The final output layer's outcome is $zs = (Xo . Es)$ (9) As a result, the GRU network's distinctive structure makes it easy to compute and train while still being effective at extracting time sequence information. 3) Time Attention The addition of an attention mechanism to the LSTM networks dramatically increased their performance and classification accuracy. a GRU that incorporates the attention mechanism model, significantly increasing real estate's accuracy. In the GRU network, $w1 - wm$represent the gate loop control unit, while $E1 - Em$ represent the output of the gate loop control unit. In the model of the temporal attention mechanism, the output of the hidden layer rs looks like this: $rs = \tan e(xxEs + a)$ (10) Where Xx stands for the weights and tanh denotes the activation function. Each output result from a GRU network is given the following weight: $bs = soft\ max\ su\ S , x\ j$ (11) The attention mechanism layer's ultimate output ps is $ps = \sum bsEs$ (12) In order to improve recognition accuracy and speed, this approach allows the network to zero in on the most important aspects of the lip sequence, which are then given greater weights.

IV. RESULTS AND DISCUSSION

A real estate entity is a physical representation of the land itself, all of its improvements, and all of the rights, interests, advantages, and obligations that come with entity ownership. Therefore, valuing real estate is an effort to give a numerical representation of the advantages and disadvantages of ownership. The job of an appraiser or assessor is to conduct a professional real estate value. The experts must link to significant economic principles that underpin how the real estate market operates in order to arrive at a value assessment. These include the laws of demand and supply, rivalry, substitution, foresight, and change. All of these concepts have a similar direct and indirect impact on a property's level of usefulness and productivity. Therefore, it may be said that the utility of real estate reflects the sum of all market influences that have an impact on a real estate parcel's value.

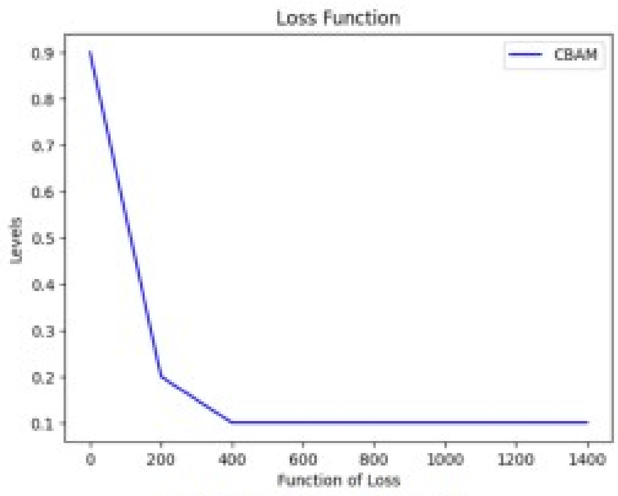


Fig. 1. Loss Function of the Models

The experimental results of the three proposed parameter sharing models demonstrate that sharing the coding and decoding layers, which accounts for the various semantic information of event relation semantic features, is the only way to integrate the features of the event relation during the coding and decoding stages. As a result, it can be used as a supplementary instrument of prediction and to establish a robust correlation between the causal relationship and timing of events. This was discovered as a result of the experimental findings shown in figure 1.

TABLE I. COMPARISION OF MODELS(%)

Models	Precision	Recall	F1-Score
GRU	0.848	0.801	0.875
LSTM	0.823	0.817	0.846
CBAM	0.854	0.739	0.962

A model selection approach (shown in Table I) led to the three models (CNN, SVM, and SVM-RFE-MRMR) as the most promising solutions. Experiments show that the SVM-RFE-MRMR algorithm model performs better than the other three models by a significant margin.

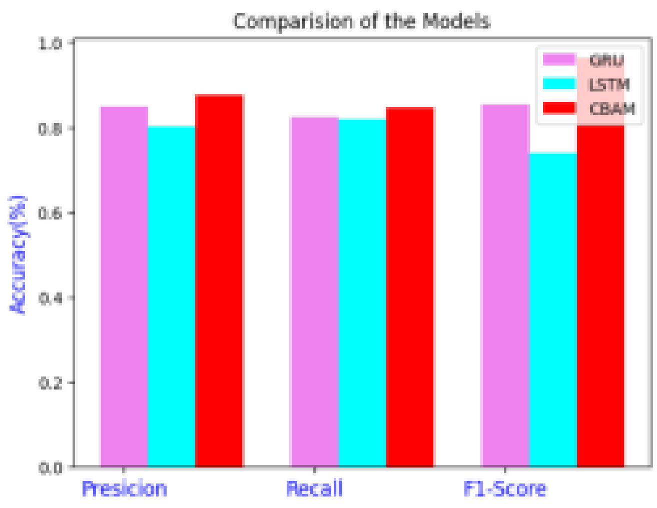


Fig. 2. Comparision of the Model

Accuracy, recall, and F1 are the metrics that are used to assess the performance of the model when it comes to determining the order of events and the relationships between them in the text. The particular values of the experimental findings are presented in Table 1, together with the comparison model and the three models that are based on the parameter sharing technique. Figure 2 depicts the accuracy, recall, and F1 of the event relation extraction performed by several models in a clearer and more concise manner. It is clear that the model that is proposed in this work has undergone significant development since it was originally conceived.

V. CONCLUSION

Big data technologies are a result of the large amount of data that is produced daily. There are hidden patterns and information in these vast volume of data. Another significant application of big data is proving to be in real estate. In this proposed approach, they will detail the

steps involved in using the plethora of real estate market data currently at our disposal to reliably predict future home price trends. It seems prudent to put money into real
estate. This system analyzes home sales data from Ames, Iowa, between 2006 and 2010 to develop models that accurately predict a home's final sale price. Linear regression, random forest, and gradient boosting models are just some of the feature selection approaches that have been used to identify the most influential elements on a home's selling price. When compared to other models,the gradient boosting model performed the best. As part of the suggested preliminary processing, data is normalized. Principal component analysis and sequential forward feature selection are used in the feature selection process. The best model for training purposes is CBAM. The proposed method is contrasted with two additional conventional techniques, LSTM and GRU.When compared to the other two conventional ways, the suggested strategy performs well.

Innovating in the field of house price prediction involves exploring novel approaches, technologies, and features that can enhance the accuracy and usability of prediction models. Here are some innovation ideas to consider:

1. **Integration of Satellite Imagery and GIS Data:**

- Utilize satellite imagery and Geographic Information System (GIS) data to extract additional features, such as neighborhood aesthetics, proximity to green spaces, or access to public transportation, which can influence house prices.

Google
Imagery ©2023 Airbus, CNES / Airbus, Maxar Technologies, U.S. Geological Survey

Google
Imagery ©2023 Airbus, CNES / Airbus, Maxar Technologies, U.S. Geological Survey

```
# Import necessary libraries
import numpy as np
import pandas as pd
import rasterio
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
```

```
# Load the dataset with house prices and satellite imagery information
# Replace this with your dataset containing features like house prices, latitude,
longitude, etc.
dataset = pd.read_csv('your_dataset.csv')

# Load satellite imagery using rasterio
# Replace 'path/to/satellite/image.tif' with the actual path to your satellite imagery
satellite_image_path = 'path/to/satellite/image.tif'
satellite_data = rasterio.open(satellite_image_path)

# Extract relevant features from satellite imagery
# For example, you could extract average pixel values in certain regions or spectral
indices
# Here, we're calculating the average pixel values across all bands
image_data = satellite_data.read()
average_pixel_values = np.mean(image_data, axis=(1, 2))

# Combine satellite features with other features from the dataset
features = pd.concat([dataset, pd.DataFrame(average_pixel_values,
columns=['band_1', 'band_2', 'band_3'])], axis=1)

# Split the data into training and testing sets
X = features.drop('house_price', axis=1)  # Assuming 'house_price' is the target
variable
y = features['house_price']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Train a Random Forest Regressor (you can use other regressors as well)
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Make predictions on the test sety_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print(f'Mean Squared Error: {mse}')

# Visualize actual vs. predicted prices
plt.scatter(y_test, y_pred)
plt.xlabel('Actual House Prices')
plt.ylabel('Predicted House Prices')
plt.title('Actual vs. Predicted House Prices')
plt.show()
```

library(tidyverse) library(tidymodels) library(textrecipes)

this is the code for **Integration of Satellite Imagery and GIS Dat.** Use this code and insert our value to predict house price

2. **Natural Language Processing (NLP) for Sentiment Analysis:**

•Exploring the data

•Data preprocessing

•Preparing the data for 10-fold cross-validation

•Lasso machine learning model fitting

•Model evaluation

Incorporate NLP techniques to analyze online reviews, social media sentiment, and news articles related to specific neighborhoods. Sentiment analysis can capture the public perception of an area, influencing house prices.

* **Data Exploration of Housing Prices**

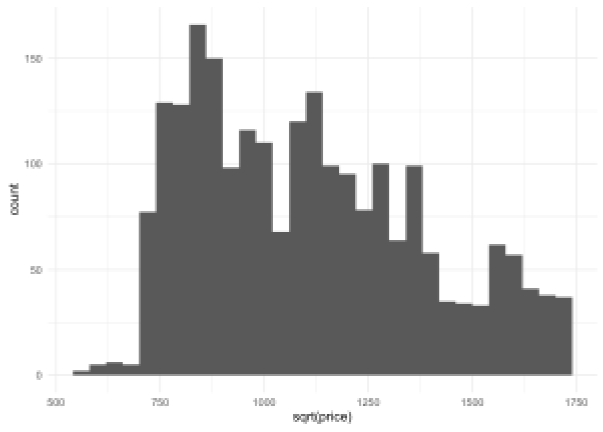
First, we will be reading in the data and then cleaning it up a bit. We will be removing outliers and potentially false data points.

```
    library(tidyverse)
    library(tidymodels)
    library(textrecipes)
    library(parallel)
    housing <- readr::read_csv("data/df.csv")
# removes houses with missing (unrealistic) year_built values
    housing <- housing[housing$year_built < 2030, ]
# remove houses that cost more than 3 million
    housing <- housing[housing$price < 3000000, ]
# remove square feet that are too low
    housing <- housing[housing$sqft > 100, ]
    housing <- housing %>%
   dplyr::select(price, description, website) %>%
   dplyr::filter(website == "buying") %>%
   na.omit() %>%
   dplyr::select(-website) %>%
   dplyr::mutate(description = stringr::str_remove_all(description,
   "[0-9]+"))
  dplyr::glimpse(housing)
```

In the end we have 2,244 rows and 2 columns. One column with the price, our response variable, and the other one is the housing descriptions. We also removed all numbers in the description as we only want to focus on the words.

```
library(tidyverse)
library(tidymodels)
library(textrecipes)
```

```
ggplot(housing, aes(x = sqrt(price))) +

geom_histogram(binwidth = 40) +
theme_minimal()
```



***Data Preprocessing For NLP and Tidymodels textrecipes**

For the preprocessing steps, we will be using the `textrecipes` package from the `tidymodels`. We will be transforming the response variable, tokenize the `description` stemming the words to their affixes and suffixes, removing all the stop words, only keeping the top 2500 words, and then translating the output into ones and zeros. I am keeping a lot of predictors because I will be using the Lasso model which will take care of the variance and shrink a lot of the predictors to zero. If I were to use another model, I would go with around 500 words.

The steps that we described above are coded below:

```
remove_words <- c(
"https", "bldg", "covid", "sqft", "bdrms",
"only.read", "included.read", "moreview", "&amp",
"baths", "bdrm", "bath", "feet", "square", "amp",
"sq.ft", "beds", "you’ll", "uniqueaccommodations.com",
"rentitfurnished.com", "it’s", "http", "below:https",
"change.a", "january", "february", "march", "april",
"may", "june", "july", "september", "october", "listing.to",
"november", "december", "note:all", "property.to", "link:http",
"www.uniqueaccomm", "www.uniqueaccomm", "change.to", "furnishedsq",
"craigslist.rental", "craigslist.professional", "ft.yaletown",
"ft.downtown")
recipe_nlp <- recipe(price ~., data = housing) %>%
recipes::step_sqrt(price) %>%
textrecipes::step_tokenize(description) %>%
textrecipes::step_stem(description) %>%
textrecipes::step_stopwords(description) %>%
textrecipes::step_stopwords(description,
custom_stopword_source = remove_words) %>%
textrecipes::step_tokenfilter(description, max_tokens = 2500) %>%
textrecipes::step_tf(description, weight_scheme = "binary") %>%
recipes::step_mutate_at(dplyr::starts_with("tf_"), fn = as.integer)
```

We also included some stop words that we thought are not contributing to a better predictability of the model.

*Cross-Validation for NLP with Tidymodels rsample

Next, we want to do 10-fold cross-validation for the best lambda and to get a sense of what the test set error will be. We do that with the `rsample` package from the `tidymodels` framework.

```
set.seed(1)
split <- rsample::initial_split(housing)
train <- rsample::training(split)
test <- rsample::testing(split)
folds <- rsample::vfold_cv(train, v = 10)
tune_spec <- parsnip::linear_reg(penalty = tune::tune(), mixture = 1) %>%
parsnip::set_engine("glmnet")
lambda_grid <- dials::grid_regular(dials::penalty(range = c(-3, 3)), levels = 100)
nlp_wflow <-
workflows::workflow() %>%
workflows::add_recipe(recipe_nlp) %>%
workflows::add_model(tune_spec)
all_cores <- parallel::detectCores()
library(doFuture)
doFuture::registerDoFuture()
cl <- makeCluster(all_cores)
plan(cluster, workers = cl)
```

I the code above, we are splitting the data set into training and testing/validation data sets. Then we are creating the 10 folds from the training data set. If you want to learn more about cross-validation, you can check out this post about cross-validation in machine learning and the bias-variance trade-off and also this one where it discusses how to do cross-validation the wrong way and right way.

We then specify the model where we want to tune the penalty. For that, we will be trying out different values with `tune::grid_values()` from the `tidymodels` framework. The mixture is set to one because we are using the lasso model. We then add the recipe we created earlier and the model specification to our workflow and then set up the model fitting in parallel.

* NLP Model Fitting and NLP Model Evaluation

```
        res <-
        nlp_wflow %>%
        tune::tune_grid(resamples = folds, grid = lambda_grid,    metril
yardstick::metric_set(yardstick::mae,
            yardstick::rsq))
            autoplot(res)
```

We are using our previously created workflow and the 10 folds to evaluate which lambda results in the best mean absolute error. I like the easy interpretation of the mean absolute error. Alternatively, we could have used rmse which penalizes more for outliers and is not as interpretable as mae.

When we plot the mae and r-squared we get this:

The mean absolute error is…

```
tune::collect_metrics(res) %>%
dplyr::filter(.metric == "mae") %>%
dplyr::arrange() %>%
.[["mean"]] %>%
.[[1]]
## [1] 149.0941
```

… 149.

Now, we are selecting the best lambda and then fitting the lasso model on all the training data.

```
best_mae <- tune::select_best(res, "mae")
final_lasso <- tune::finalize_workflow(
nlp_wflow,
best_mae
)
final_lasso
## ══ Workflow ════════════════════════════════════════════════════════════════════════════════
## Preprocessor: Recipe
## Model: linear_reg()
##
## ── Preprocessor ────────────────────────────────────────────────────────────────────
## 8 Recipe Steps
##
## ● step_sqrt()
## ● step_tokenize()
## ● step_stem()
## ● step_stopwords()
## ● step_stopwords()
## ● step_tokenfilter()
## ● step_tf()
## ● step_mutate_at()
```

```
##
## ── Model
────────────────────────────────────────────────────────────────────────────
────────────────────
## Linear Regression Model Specification (regression)
##
## Main Arguments:
## penalty = 2.8480358684358
## mixture = 1
##
## Computational engine: glmnet
housing_final <- tune::last_fit(
final_lasso,
split,
metrics = yardstick::metric_set(yardstick::mae,
yardstick::rsq)
)
tune::collect_metrics(housing_final)
## # A tibble: 2 x 4
## .metric .estimator .estimate .config
## <chr> <chr> <dbl> <chr>
## 1 mae standard 146. Preprocessor1_Model1
## 2 rsq standard 0.528 Preprocessor1_Model1
```

We have a mean absolute error of 146 and an r-squared of around 146.

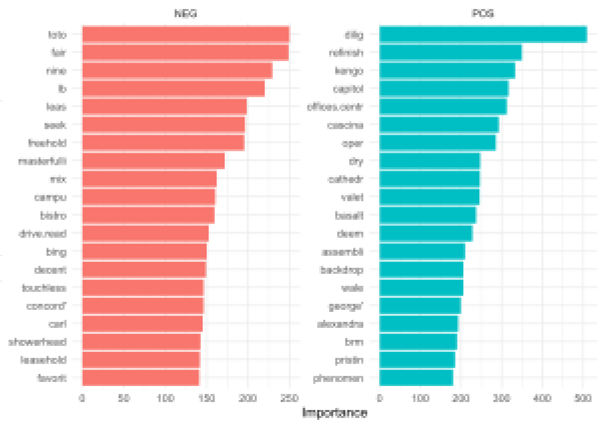
***Visualizing NLP Important Words**

Next, we will have a look at what predictors are most important in our NLP Lasso model.

```
housing_vip <- housing_final %>%
pull(.workflow) %>%
.[[1]] %>%
workflows::pull_workflow_fit() %>%
vip::vi()
housing_vip %>%
dplyr::mutate(Variable = stringr::str_remove(Variable, "tf_description_")) %>%
dplyr::group_by(Sign) %>%
dplyr::slice_max(abs(Importance), n = 20) %>%
ungroup() %>%
mutate(
Variable = fct_reorder(Variable, Importance)
) %>%
ggplot(aes(Importance, Variable, fill = Sign)) +
geom_col() +
```

library(tidyverse)
library(tidymodels)
library(textrecipes)

```
facet_wrap(~Sign, scales = "free") +
labs(y = NULL) +
theme_minimal() +
theme(legend.position = "none")
```

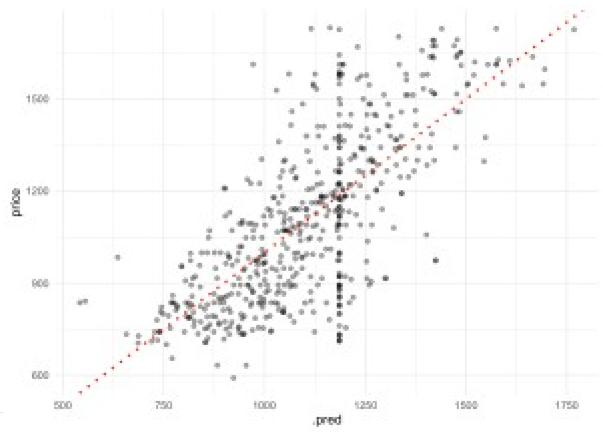


3. **Temporal Analysis and Market Trends:**

- Implement temporal
- The words you can see above are the top 20 words that are affecting the price of a real estate properly most positively and negatively.

Next, we will have a look at our predictions.

```
tune::collect_predictions(housing_final) %>%
ggplot(aes(x = .pred, y = price)) +
geom_point(alpha = 0.4) +
geom_abline(slope = 1, linetype = "dotted",
col = "red", size = 1) +
theme_minimal()
```



- analysis to capture evolving market trends over time. This can include seasonal variations, economic indicators, and other time-dependent factors that impact house prices.

We can see that for lower property prices, our model is overestimating the prices more and for higher property prices the model is predicting them more to be less than the actual value.

When we look at the mean absolute error, then we can see that our model is off $332,929 on average. For a three million house that would be 10% off which is not bad but for properties priced at $500,000 that's pretty bad. Overall, our model is decent at best, however, given that we only used words and no bigrams or anything else it is not too bad. Maybe some more accuracy can be achieved by some more feature engineering.

```
tune::collect_predictions(housing_final) %>%
dplyr::mutate(mae = abs(.pred^2 - price^2)) %>%
.[["mae"]] %>%
mean()
## [1] 332929.3
```

4. **Dynamic Pricing Models:**
    - Explore dynamic pricing models that consider real-time market conditions, economic changes, and demand fluctuations. This can provide more adaptive and responsive pricing predictions.
5. **Incorporation of Economic Indicators:**
    - Integrate a broader range of economic indicators, such as inflation rates, interest rates, and employment statistics, into the prediction model to capture the macroeconomic factors influencing the housing market.
6. **Explainable AI (XAI):**
    - Implement AI models with explainability features, allowing users to understand and trust the model's predictions. This is crucial for gaining acceptance and transparency in the real estate industry.
7. **Blockchain for Property Transactions:**
    - Explore the use of blockchain technology for secure and transparent property transactions. This can enhance the reliability of historical transaction data used in prediction models.
8. **Augmented Reality (AR) for Virtual Property Tours:**
    - Integrate AR technology to provide virtual property tours, allowing users to explore properties remotely. This feature can enhance the user experience for potential buyers.
9. **Collaborative Filtering for Personalized Predictions:**
    - Implement collaborative filtering techniques, similar to those used in recommendation systems, to provide personalized predictions based on user preferences and historical interactions with property listings.
10. **Climate and Environmental Impact Analysis:**

- Include climate and environmental impact factors in the prediction model, such as susceptibility to natural disasters or the property's carbon footprint. This could appeal to environmentally conscious buyers.

11. **Smart Home Features as Predictors:**

    - Consider the inclusion of features related to smart home technology, energy efficiency, and sustainability, as these factors are increasingly influencing property values.

12. **Gamification for User Engagement:**

    - Introduce gamification elements to engage users in the prediction process. For example, users could predict future house prices, and the system could provide feedback on the accuracy of their predictions.

Remember to thoroughly test and validate any new features or technologies before integrating them into the production environment. Additionally, keeping an eye on emerging technologies and industry trends can help you stay at the forefront of innovation in house price prediction.

3.House price pridiction using loading and preprocessing the dataset

Loading and preprocessing the dataset are crucial steps in any machine learning project, including house price prediction. Here, I'll provide a basic example using Python and popular libraries like pandas and scikit-learn

Introduction

First of all, the challenge goal is to predict the final price of houses based on past house sales.

To achieve this we have a dataset with **80 explanatory variables** describing houses from the city of Ames (Iowa) and the sale price of each of these houses for the target value to predict.

Let's dive into the data !

3.1) Loading data

First, let's import some common Python libraries that we will use all along this challenge.

```
import os
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
pd.set_option('display.max_columns', 80)
# plot libs
import seaborn as sns
sns.set(style="white", color_codes=True)
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (12,8)
plt.rcParams["axes.labelsize"] = 16
```

Then, we just read the CSV files containing our training set and datasets and print their size.

```
train_data = pd.read_csv("../input/train.csv", encoding = "ISO-8859-1")
test_data = pd.read_csv("../input/test.csv", encoding = "ISO-8859-1")
print("The training set shape is : %s" % str(train_data.shape))
print("The test set shape is : %s" % str(test_data.shape))
```

```
The training set shape is : (1460, 81)
The test set shape is : (1459, 80)
```

We have 1460 rows representing our houses in our training set, and one less in the test set. Notice that we have one missing column on our test set, for sure it's the SalePrice ! The one we have to predict.

Let's preview the 10 first rows of the training set.

In [3]:

```
train_data.head(n=10)
```

10 rows × 81 columns

Linkcode (Kaggle. https://kaggle.com/competitions/house-prices-advanced-regression-techniques)

3.2) Imputing missing values

```
train_and_test_data = pd.concat([train_data, test_data])
sum_result = train_and_test_data.isna().sum(axis=0).sort_values(ascending=False)
missing_values_columns = sum_result[sum_result > 0]
print('They are %s columns with missing values : \n%s' %
(missing_values_columns.count(), [(index, value) for (index, value) in
missing_values_columns.iteritems()]))
```

```
They are 35 columns with missing values :
[('PoolQC', 2909), ('MiscFeature', 2814), ('Alley', 2721), ('Fence', 2348),
('SalePrice', 1459), ('FireplaceQu', 1420), ('LotFrontage', 486),
('GarageFinish', 159), ('GarageCond', 159), ('GarageQual', 159), ('GarageYrBlt',
159), ('GarageType', 157), ('BsmtCond', 82), ('BsmtExposure', 82), ('BsmtQual',
81), ('BsmtFinType2', 80), ('BsmtFinType1', 79), ('MasVnrType', 24),
('MasVnrArea', 23), ('MSZoning', 4), ('BsmtFullBath', 2), ('BsmtHalfBath', 2),
('Utilities', 2), ('Functional', 2), ('Electrical', 1), ('Exterior2nd', 1),
('KitchenQual', 1), ('Exterior1st', 1), ('GarageCars', 1), ('TotalBsmtSF', 1),
('GarageArea', 1), ('BsmtUnfSF', 1), ('BsmtFinSF2', 1), ('BsmtFinSF1', 1),
('SaleType', 1)]
```

Of course the missing SalePrice are the one we try to predict as it match the number of rows in the test set.

For all of these columns, we will impute the missing values with this strategy :

- For categorical variable : "NA", for example we can assume that a house with no information for the pool quality means that the house have no house
- For numerical variable : median value of the variable

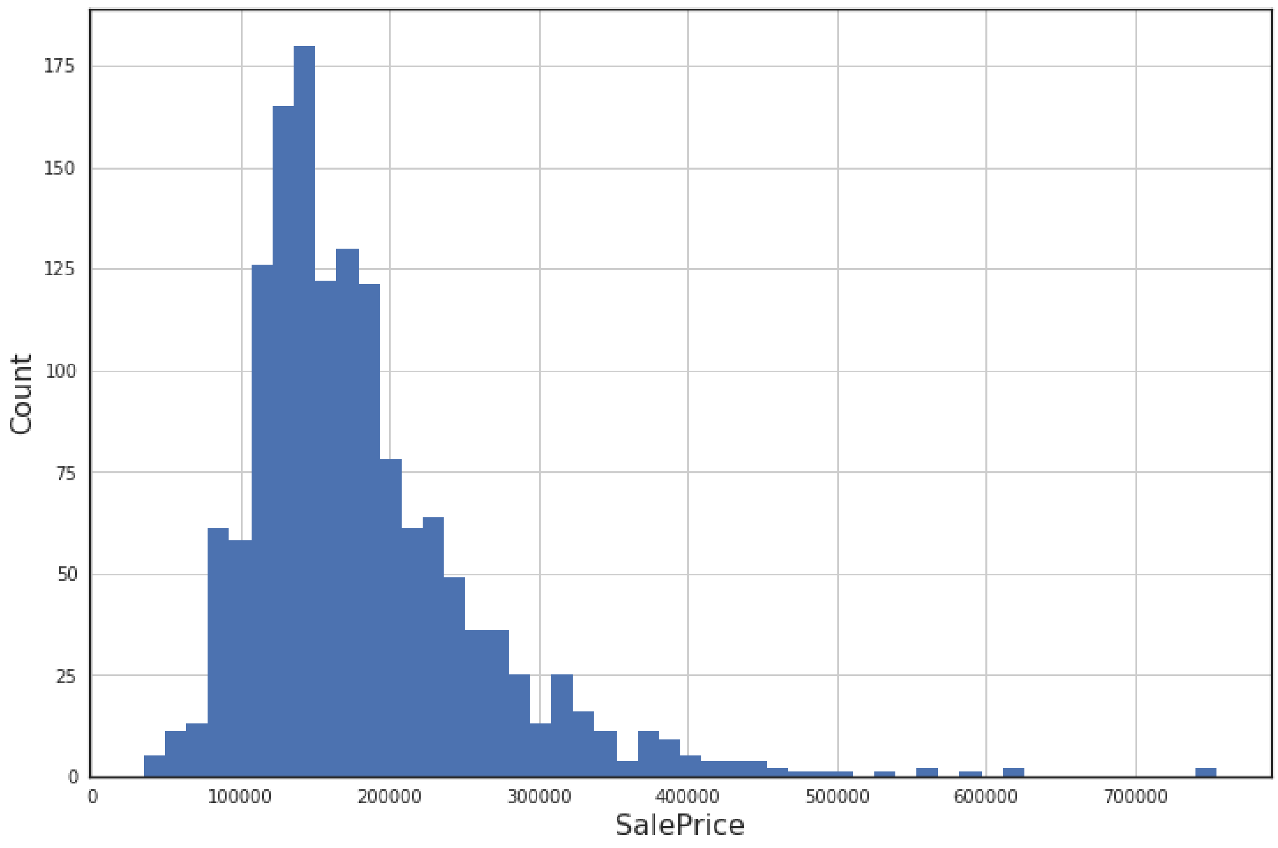
```
def impute_missing_values(train_data):
    dataset = train_data
    dataset["PoolQC"].fillna("NA", inplace=True)
    dataset["MiscFeature"].fillna("NA", inplace=True)
    dataset["Alley"].fillna("NA", inplace=True)
    dataset["Fence"].fillna("NA", inplace=True)
    dataset["FireplaceQu"].fillna("NA", inplace=True)
    dataset["LotFrontage"].fillna(dataset["LotFrontage"].median(), inplace=True)
    dataset["GarageType"].fillna("NA", inplace=True)
    dataset["GarageQual"].fillna("NA", inplace=True)
    dataset["GarageCond"].fillna("NA", inplace=True)
    dataset["GarageFinish"].fillna("NA", inplace=True)
    dataset["GarageYrBlt"].fillna(dataset["GarageYrBlt"].median(), inplace=True)
    dataset["BsmtExposure"].fillna("NA", inplace=True)
    dataset["BsmtFinType2"].fillna("NA", inplace=True)
    dataset["BsmtQual"].fillna("NA", inplace=True)
    dataset["BsmtCond"].fillna("NA", inplace=True)
    dataset["BsmtFinType1"].fillna("NA", inplace=True)
    dataset["MasVnrArea"].fillna(dataset["MasVnrArea"].median(), inplace=True)
    dataset["MasVnrType"].fillna("None", inplace=True)
    dataset["Electrical"].fillna("SBrkr", inplace=True)  # SBrkr is the most 
common value for 1334 houses
    dataset["BsmtQual"].fillna("NA", inplace=True)
    dataset["MSZoning"].fillna("TA", inplace=True)
    dataset["BsmtFullBath"].fillna(0, inplace=True)
    dataset["BsmtHalfBath"].fillna(0, inplace=True)
    dataset["Utilities"].fillna("AllPub", inplace=True)
    dataset["Functional"].fillna("Typ", inplace=True)
    dataset["Electrical"].fillna("SBrkr", inplace=True)
    dataset["Exterior2nd"].fillna("VinylSd", inplace=True)
    dataset["KitchenQual"].fillna("TA", inplace=True)
    dataset["Exterior1st"].fillna("VinylSd", inplace=True)
    dataset["GarageCars"].fillna(0, inplace=True)
    dataset["GarageArea"].fillna(dataset["GarageArea"].median(), inplace=True)
    dataset["TotalBsmtSF"].fillna(dataset["TotalBsmtSF"].median(), inplace=True)
    dataset["BsmtUnfSF"].fillna(dataset["BsmtUnfSF"].median(), inplace=True)
    dataset["BsmtFinSF2"].fillna(dataset["BsmtFinSF2"].median(), inplace=True)
    dataset["BsmtFinSF1"].fillna(dataset["BsmtFinSF1"].median(), inplace=True)
    dataset["SaleType"].fillna("WD", inplace=True)
    return dataset

train_data = impute_missing_values(train_data)
test_data = impute_missing_values(test_data)
```

3.3) Data exploration

Before we focus on the explanatory variables, we can begin the data exploration by analizing the SalePrice target value.

3.3.1) The target value : SalePrice

```
plt.hist(train_data["SalePrice"], 50)
plt.xlabel("SalePrice")
plt.ylabel("Count")
plt.grid(True)
plt.show()
```



This histogram show that a majority of the prices doesn't exceed 230 000$. Using the `describe()` function we can have basics stats on this SalePrice column.

```
train_data["SalePrice"].describe()
```

Out[7]:

```
count       1460.000000
mean      180921.195890
std        79442.502883
min        34900.000000
25%       129975.000000
50%       163000.000000
```

```
75%      214000.000000
max      755000.000000
Name: SalePrice, dtype: float64
```

3.3.2) Important numeric variables : OverallQual and GrLivArea

With 80 variables, we sure have a lot of work to understand them and do some feature engineering in order to feed them in a machine learning algorithm. We have categorical variables (oridinal and non ordinal) and also numerical variables. Numerical variables is less complicated to analyze and transform, so we will begin with this kind of variables.

```
numeric_cols = list(train_data.select_dtypes(include=[np.number]))
numeric_cols.remove('Id')
numeric_cols.remove('MSSubClass')
print("Here are the %s numeric variables : \n %s" % (len(numeric_cols),
numeric_cols))
```

```
Here are the 36 numeric variables :
 ['LotFrontage', 'LotArea', 'OverallQual', 'OverallCond', 'YearBuilt',
'YearRemodAdd', 'MasVnrArea', 'BsmtFinSF1', 'BsmtFinSF2', 'BsmtUnfSF',
'TotalBsmtSF', '1stFlrSF', '2ndFlrSF', 'LowQualFinSF', 'GrLivArea',
'BsmtFullBath', 'BsmtHalfBath', 'FullBath', 'HalfBath', 'BedroomAbvGr',
'KitchenAbvGr', 'TotRmsAbvGrd', 'Fireplaces', 'GarageYrBlt', 'GarageCars',
'GarageArea', 'WoodDeckSF', 'OpenPorchSF', 'EnclosedPorch', '3SsnPorch',
'ScreenPorch', 'PoolArea', 'MiscVal', 'MoSold', 'YrSold', 'SalePrice']
```

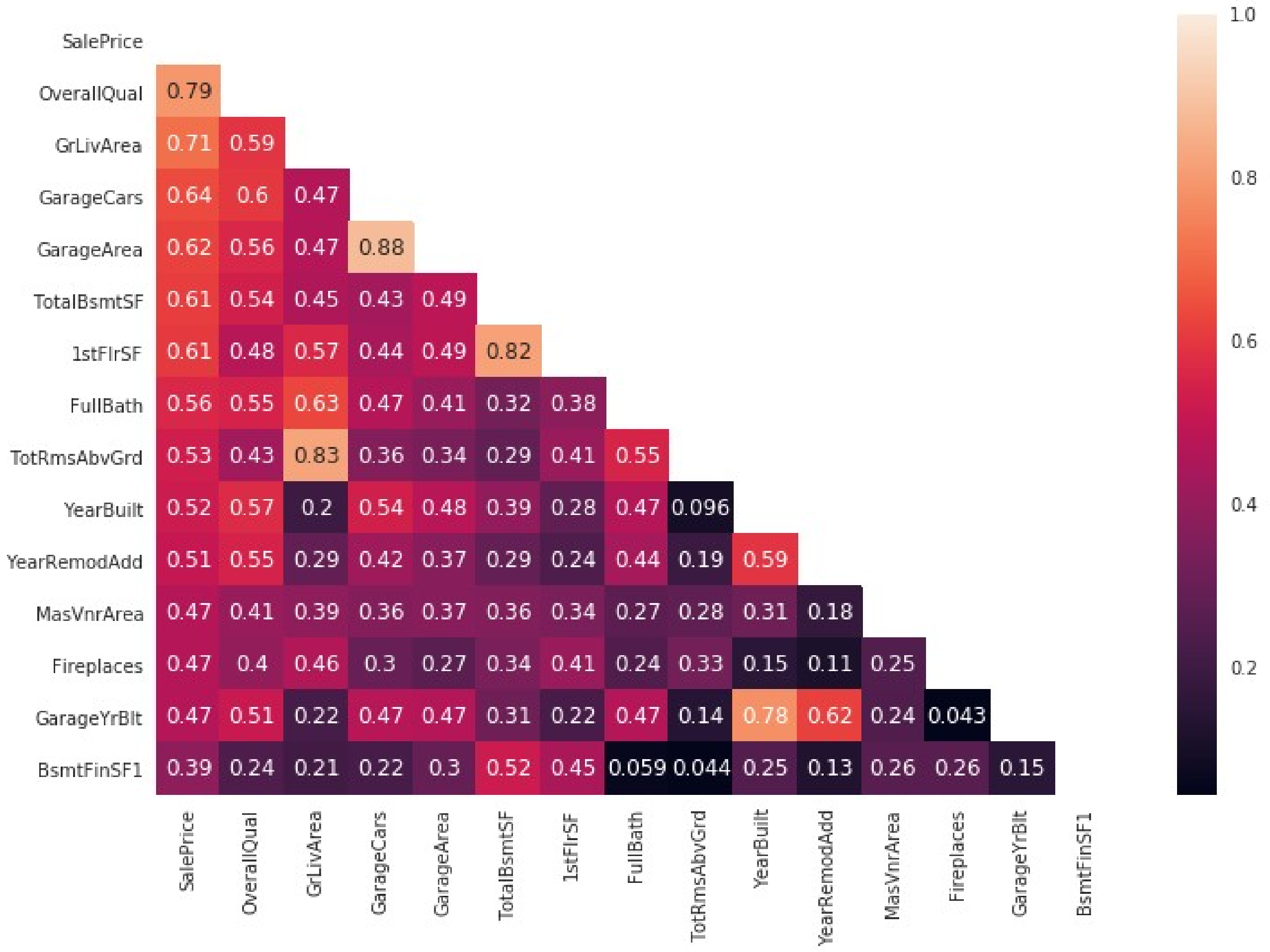
We exluded from this list the `Id` column which is an unique identifier for the houses. Obviously it's not going to help in the price prediction. The `MSSubClass` is not really a numeric variables, as explained in `data_description.txt` file it's in fact a categorical variable. We will process it later to transform it in a categorical variable.

Let's see the correlation with these numerical variables and the target value. We will just keep the 15 first variables more correlated.

Code:

```
numerical_values =
train_data[list(train_data.select_dtypes(include=[np.number]))]
# Get the more correlated variables by sorting in descending order for the
SalePrice column
```

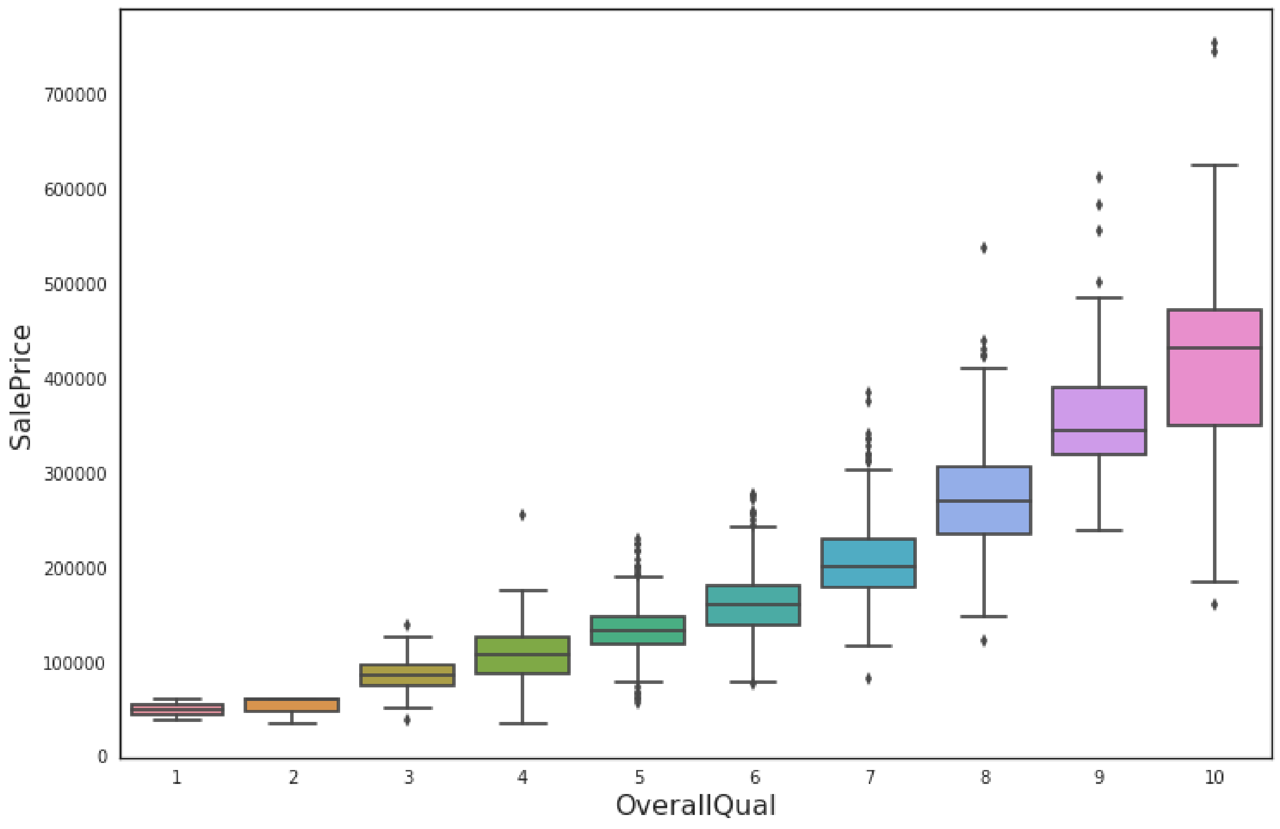
```
ix = numerical_values.corr().sort_values('SalePrice', ascending=False).index
df_sorted_by_correlation = numerical_values.loc[:, ix]
# take only the first 15 more correlated variables
fifteen_more_correlated = df_sorted_by_correlation.iloc[:, :15]
corr = fifteen_more_correlated.corr()
mask = np.zeros_like(corr)
mask[np.triu_indices_from(mask)] = True
with sns.axes_style("white"):
    # display a correlation heatmap
    ax = sns.heatmap(corr, mask=mask, annot=True)
```



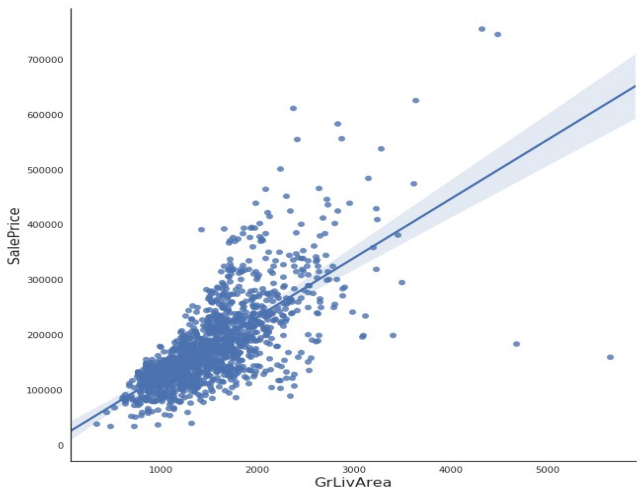
The overall quality and the above grade living area are the most correlated variables to the SalePrice. We can take a closer look to them by ploting their relation to the SalePrice.

code

```
sns.boxplot(x="OverallQual", y="SalePrice", data=train_data[['OverallQual',
'SalePrice']])
plt.show()
```



```
sns.pairplot(data=train_data,     x_vars=['GrLivArea'],     y_vars=['SalePrice'],
size=9, kind='reg')
plt.show()
```



With these two plots, we can clearly see a linear correlation, so the Overall quality notation and the Above grade (ground) living area square feet are very important in the determination of the SalePrice.

3.3.3) Important categorical variables : MSSubClass and Neighborhood

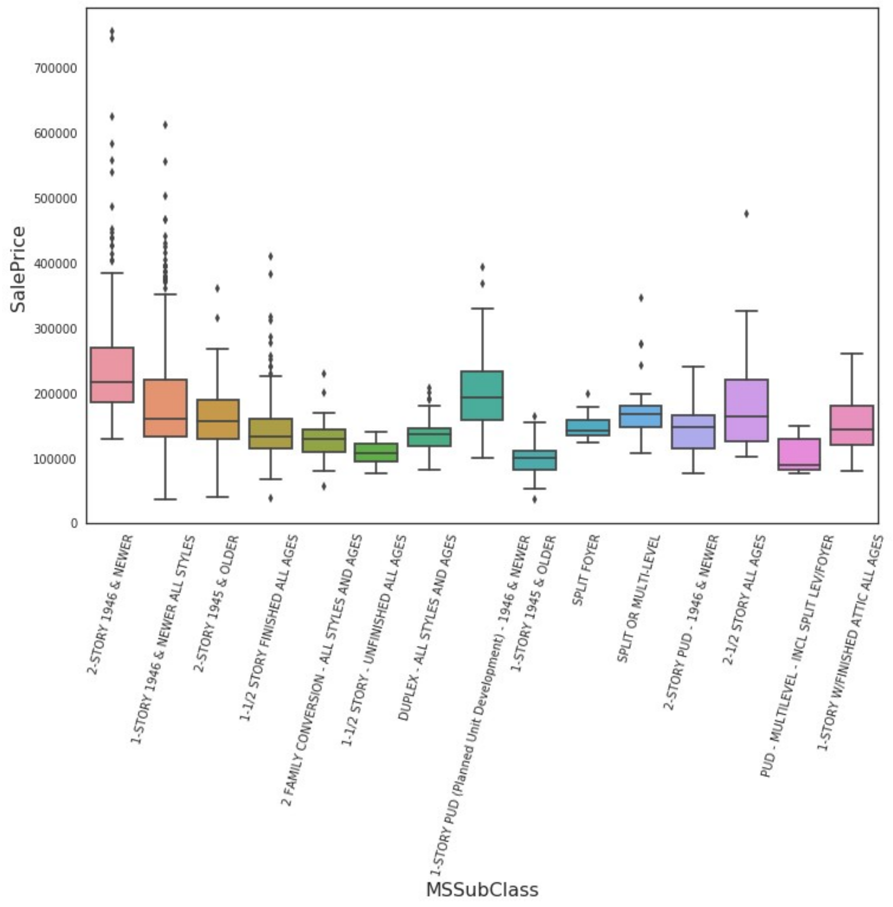
First we will transform the `MSSubClass` column and replace the numerical values by their labels.

code

```
train_data.replace({'MSSubClass': {
    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",
}}, inplace=True)
```

With the box plot below, we can see the importance of the type of dwelling involved in the sale (MSSubClass variable) for the price determination.

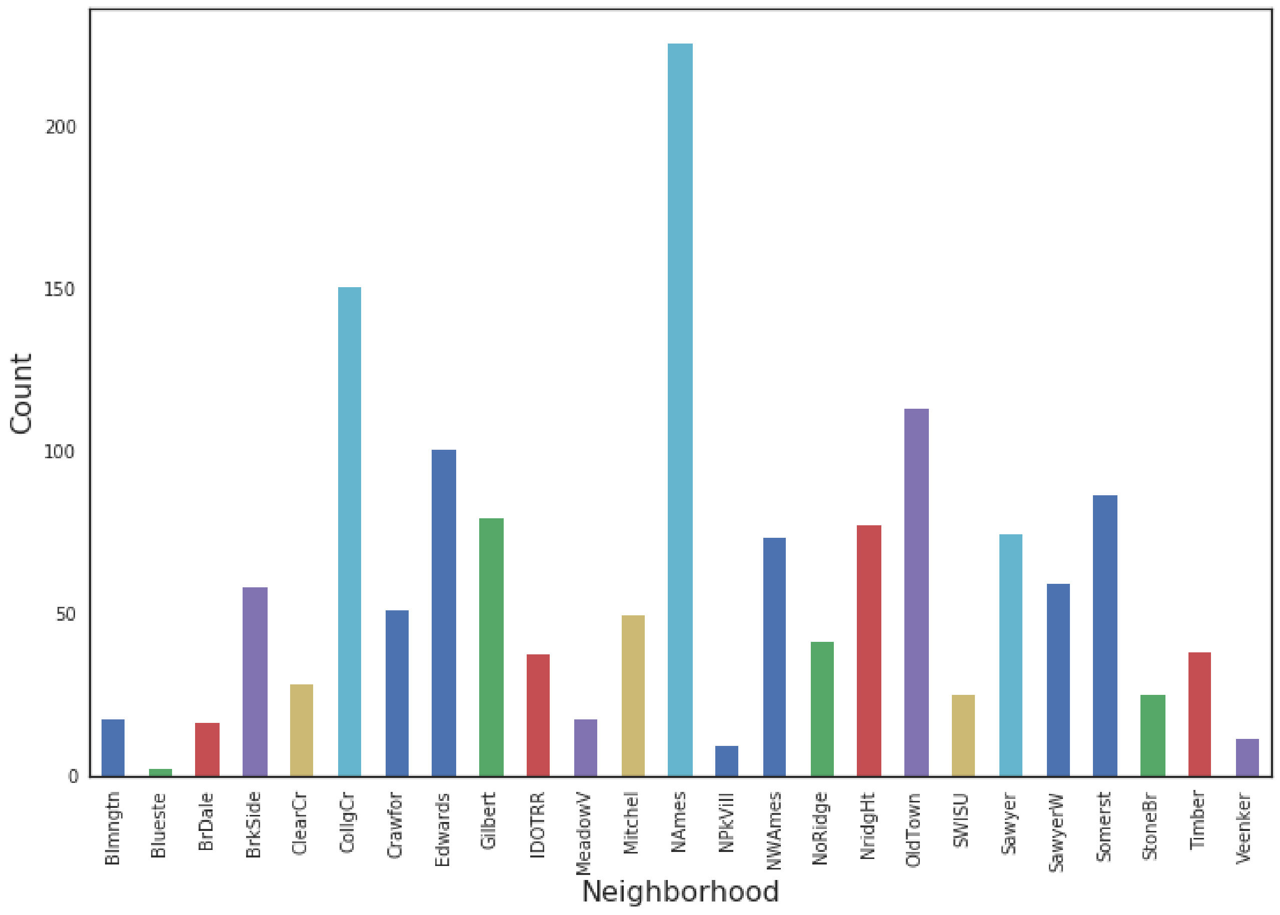
```
g = sns.boxplot(x="MSSubClass", y="SalePrice", data=train_data)
for item in g.get_xticklabels():
    item.set_rotatvion(75)
```



The first plot is an histogram with the SalePrice median value of each Neighborhood categories. You can see that some Neighborhood values tend to determine a low or high price. The second plot show the number of houses reprensented in each category.

code

```
     neighborhood_median_plot =
train_data.groupby('Neighborhood['SalePrice'].median().plot(kind='bar')
neighborhood_median_plot.set_ylabel('SalePrice')
 h = neighborhood_median_plot.axhline(train_data['SalePrice'].mean())
```



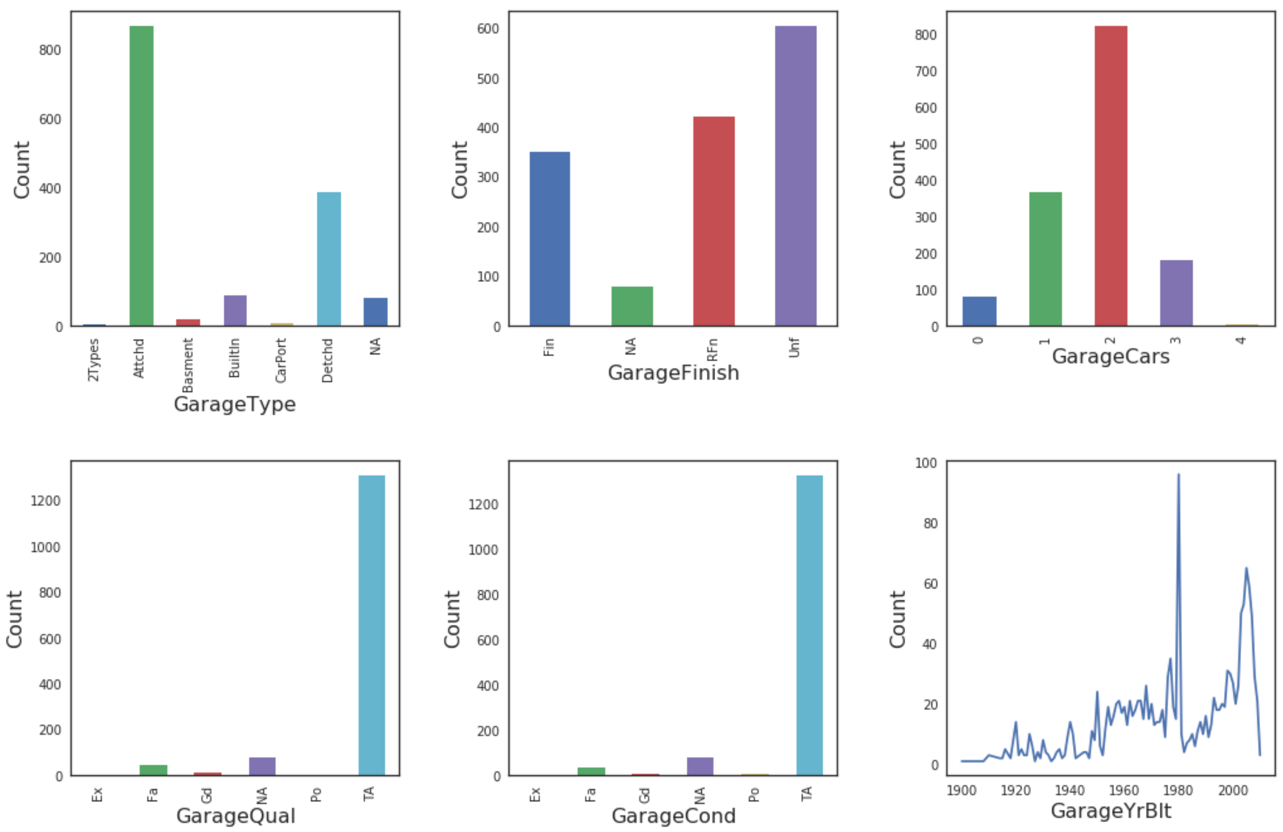
3.3.4) Other Important variables : Garage variables

There are 7 variables around the house garage. As we already seen in the correlation heat map of the first 15 numeric variables, `GarageCars` and `GarageArea` are in the third and fourth positions. We may keep some of these 7 variables to predict the `SalePrice`.**

code

```
fig, axes = plt.subplots(nrows=2, ncols=3, squeeze=True)
figsize = (15, 10)
train_data.groupby("GarageType")["GarageType"].count().plot(kind='bar',
ax=axes[0][0], figsize=figsize).set_ylabel('Count')
train_data.groupby("GarageFinish")["GarageFinish"].count().plot(kind='bar',
ax=axes[0][1], figsize=figsize).set_ylabel('Count')
```

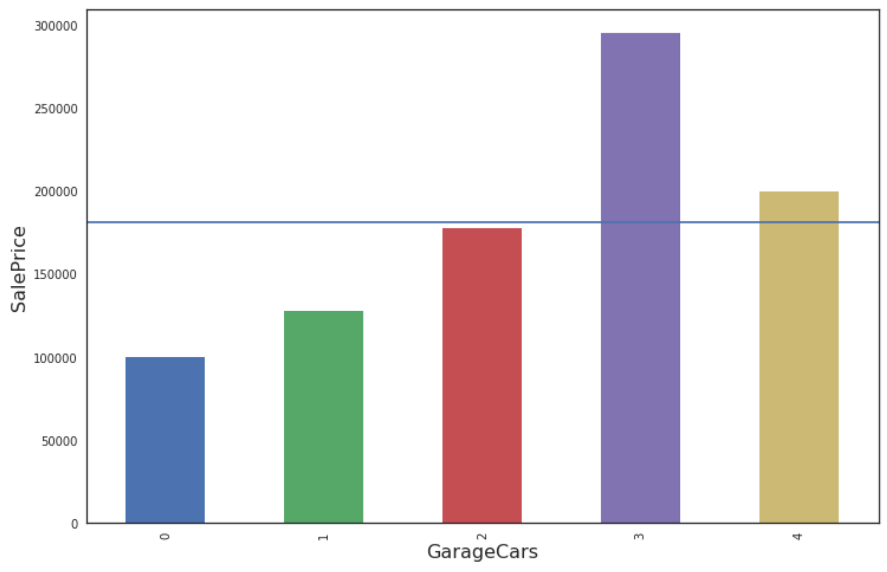
```
train_data.groupby("GarageCars")["GarageCars"].count().plot(kind='bar',
ax=axes[0][2], figsize=figsize).set_ylabel('Count')
train_data.groupby("GarageQual")["GarageQual"].count().plot(kind='bar',
ax=axes[1][0], figsize=figsize).set_ylabel('Count')
train_data.groupby("GarageCond")["GarageCond"].count().plot(kind='bar',
ax=axes[1][1], figsize=figsize).set_ylabel('Count')
train_data.groupby("GarageYrBlt")["GarageYrBlt"].count().plot(kind='line',
ax=axes[1][2], figsize=figsize).set_ylabel('Count')
fig.tight_layout(pad=3, w_pad=3, h_pad=3)
```



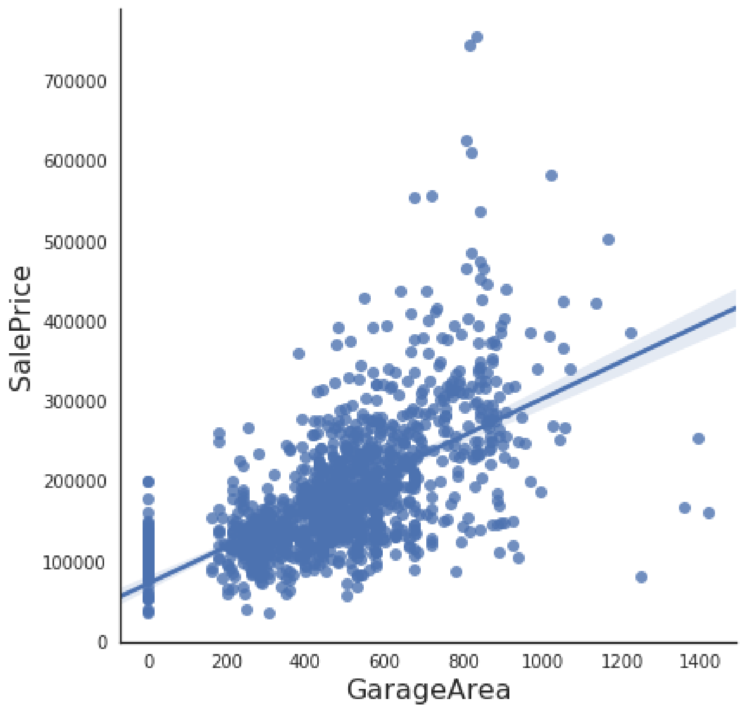
The GarageQual and GarageCond variables are not well represented as the TA (meaning Average/Typical) value dominate largely. The GarageCars is maybe a good candidate to predict the house SalePrice, it's the fourth more correlated variable and as we can see with the plot below the price varies with the garage car capacity.

code

```
garage_cars_median_plot = train_data.groupby('GarageCars')
['SalePrice'].median().plot(kind='bar')
garage_cars_median_plot.set_ylabel('SalePrice')
h = garage_cars_median_plot.axhline(train_data['SalePrice'].mean())
```



```
g = sns.pairplot(data=train_data, x_vars=['GarageArea'], y_vars=['SalePrice'],
size=6, kind='reg')
```



We can see a correlation between the `GarageArea` and the `SalePrice`, but we may not keep it to predict as we already have a better correlated value representing the area of the house : `GrLivArea`.

3.4) Feature engineering

Machine learning models ask for numeric values, so we need to transform categorical variables. We have to determine for each categorical variables which is ordinal and non-ordinal. We may also create new variables from existing ones (like for example regroup variables in the same subject with a new variable), but for now we will try without it.

unfold_moreShow hidden code

3.4.1) Custom numerical mapping for ordinal categorical variables

We have a lot of ordinal variables like for example `ExterQual` (quality of the material on the exterior)

Here are the possible values of ExterQual :

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

In this case, it's quite simple, we replace the categorical following by mapping numerical numbers like below :

```
result = pd.DataFrame()
result['ExterQual'] = train_data['ExterQual'].replace({"Ex": 5, "Gd": 4, "TA":
3, "Fa": 2, "Po": 1, "NA": 0})
result.head()
```

Out[20]:

	ExterQual
0	4
1	3
2	4
3	3
4	4

For some variables, it's difficult to know directly if there are ordinal or non ordinal. It's the case with `MasVnrType` (Masonry veneer type)

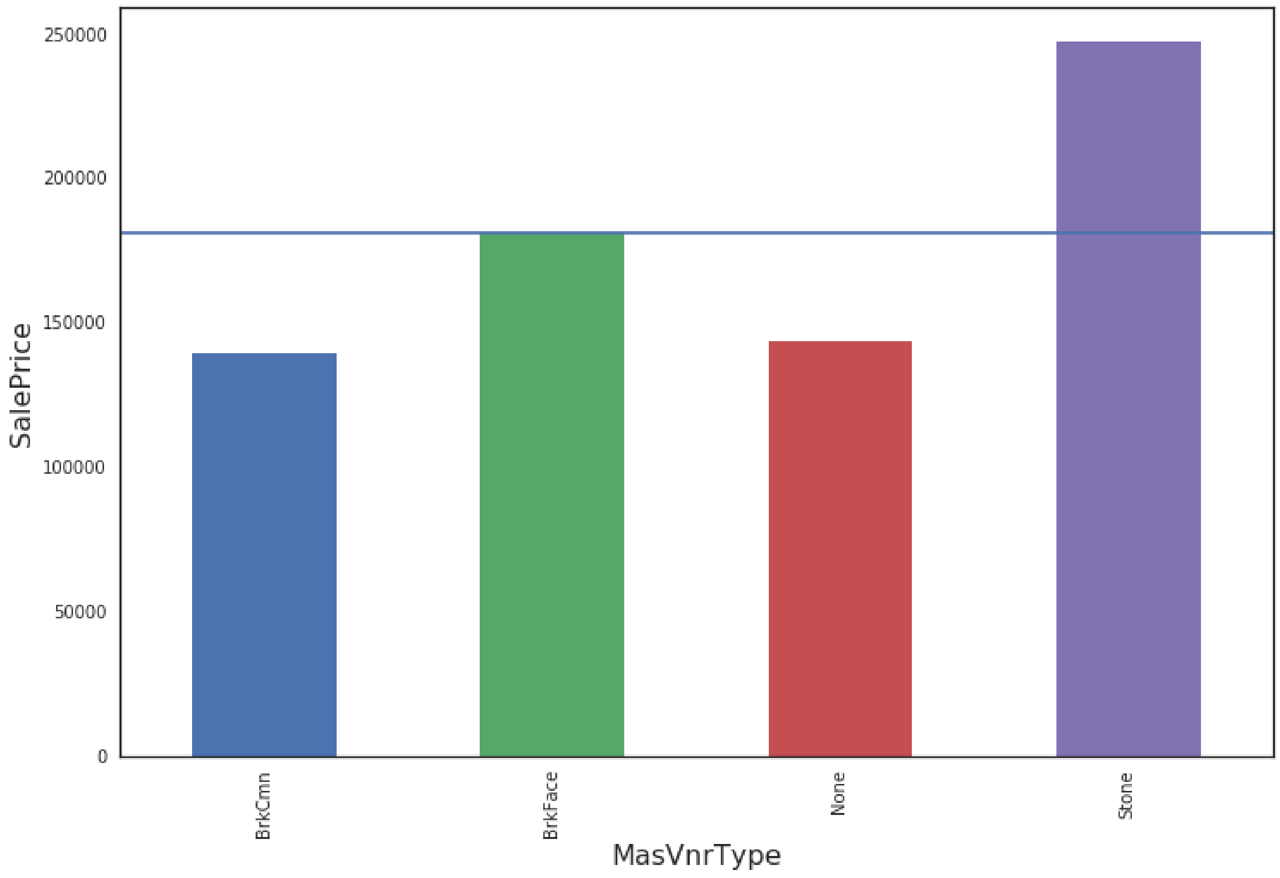
Here are the possible values of `MasVnrType` :

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

We can do a quick vizualisation of the median `SalePrice` of each value of `MasVnrType` to determine if there are a sense of ordinality.

code

```
garage_cars_median_plot = train_data.groupby('MasVnrType')
['SalePrice'].median().plot(kind='bar')
garage_cars_median_plot.set_ylabel('SalePrice')
h = garage_cars_median_plot.axhline(train_data['SalePrice'].mean())
```



We can see that Common brick and None values often means house with low `SalePrice`, Brick face is a bit more expensive and the Stone value seems to be the more expensive. We can assume that the `MasVnrType` is ordinal.

We can apply this transformation :

```
result = pd.DataFrame()
```

```
result['MasVnrType'] = train_data['MasVnrType'].replace({'None': 0, 'BrkCmn': 0,
'BrkFace': 1, 'Stone': 2})
result.head(n=8)
```

Out[22]:

	MasVnrType
0	1
1	0
2	1
3	0
4	1
5	0
6	2
7	2

3.4.2) One hot encoding for non ordinal categorical variables

For non ordinal variables like for example `LotConfig` variable :

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

We can use one hot encoding to create as many columns with 0 and 1 as variable values.

```
result = pd.get_dummies(train_data['LotConfig'], prefix='LotConfig')
result.head()
```

Out[23]:

	LotConfig_Corner	LotConfig_CulDSac	LotConfig_FR2	LotConfig_FR3	LotConfig_Inside
0	0	0	0	0	1
1	0	0	1	0	0
2	0	0	0	0	1
3	1	0	0	0	0
4	0	0	1	0	0

code

You can see at the beginning of the part 5) all the transformation I applied to the train data set by clicking on the code button.

3.5) Modeling

```
y_train = train_data['SalePrice']
X_train = transform_variables(train_data)

X_test = test_data
impute_missing_values(X_test)
X_test = transform_variables(X_test)

for col_name in list(X_train.columns):
    if col_name not in X_test.columns:
        X_test[col_name] = 0

# need to investigate why X_test got extra columns compare to X_train
test_cols = list(X_test.columns)
train_cols = list(X_train.columns)
def Diff(li1, li2):
    return (list(set(li1) - set(li2)))
X_test = X_test.drop(Diff(test_cols, train_cols), axis=1)

predictor_cols = [col for col in X_train
                  if col != 'SalePrice'
                 ]

print(str(X_test.shape) + " should be similar to " + str(X_train.shape))
```

```
(1459, 244) should be similar to (1460, 244)
```

5.1) Lasso regression model

In this step we try to feed a Lasso regression model with all of our variables.

```
from math import floor
from sklearn.preprocessing import MinMaxScaler

# filter some variables under represented
# number_of_ones_by_cols = X_train.astype(bool).sum(axis=0)
# less_than_ten_ones_cols = number_of_ones_by_cols[number_of_ones_by_cols <
10].keys()
# X_train = X_train.drop(list(less_than_ten_ones_cols), axis=1)
# X_test = X_test.drop(list(less_than_ten_ones_cols), axis=1)

# cols_to_scale = ['GrLivArea', 'TotalBsmtSF', 'GarageArea']
# scaler = MinMaxScaler()
# X_train[cols_to_scale] = scaler.fit_transform(X_train[cols_to_scale])
# X_test[cols_to_scale] = scaler.fit_transform(X_test[cols_to_scale])

from sklearn import linear_model

clf = linear_model.Lasso(alpha=1, max_iter=10000)
clf.fit(X_train[predictor_cols], y_train)

y_pred = clf.predict(X_test[predictor_cols])

print(clf.intercept_)
print(y_pred)
my_submission = pd.DataFrame({'Id': X_test.Id, 'SalePrice': y_pred})
my_submission.to_csv('lasso.csv', index=False)
```

```
-876822.4562413673
[107977.86694599 154548.86532966 173678.73739872 ... 161251.47432591
 113563.30300675 221340.38954469]
```

For now it results with a RMSLE score of around *0.18*, which can be improve. In the future I will try the following steps to get a more accurate result !

Future improvements

1. Check multicollinearity (with a linear regression model like random forest for example)
2. Removing outliers
3. Feature engineering : create new variables
4. Preprocessing predictor variables (feature scaling etc.)
5. Analize variable and target value skewness and normalize it
6. Try to use the Xgboost method

Loading preprocessing the dataset for house price prediction

Preprocessing the dataset is a crucial step in building a machine learning model for house price prediction. Proper preprocessing helps in handling missing data, dealing with categorical variables, and scaling numerical features. Here's a general guide on how to preprocess a dataset for house price prediction:

1.Import Libraries: Start by importing the necessary libraries.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

2. **Load the Dataset:** Load your dataset into a pandas DataFrame. Make sure the dataset is in a format that pandas can read, such as CSV or Excel.

```
# Assuming your dataset is in a CSV file
dataset_path = 'path/to/your/dataset.csv'
df = pd.read_csv(dataset_path)
```

3. **Explore the Dataset:** Take a quick look at the dataset to understand its structure and contents.

```
print(df.head())
print(df.info())
```

4. **Handle Missing Data:** Check for missing values in the dataset and handle them appropriately (e.g., by removing or imputing missing values).

```
# Drop rows with missing values
df = df.dropna()

# Or impute missing values
# df = df.fillna(df.mean())  # Use mean imputation as an example
```

5. **Feature Selection:** Identify the features (independent variables) and the target variable (dependent variable).

```
# Assuming 'price' is the target variable
X = df.drop('price', axis=1)
y = df['price']
```

6. **Split the Dataset:** Split the dataset into training and testing sets to evaluate the model's performance.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

7. **Feature Scaling:** Standardize or normalize the features to ensure they are on a similar scale.

```
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

8. **Save the Preprocessed Data (Optional):** Optionally, you can save the preprocessed data for future use.

```
# Save the preprocessed data
preprocessed_data_path = 'path/to/preprocessed_data.csv'
```

```
preprocessed_df = pd.DataFrame(data=X_train_scaled, columns=X.columns)
preprocessed_df['price'] = y_train
preprocessed_df.to_csv(preprocessed_data_path, index=False)
```

Now you have a preprocessed dataset that you can use for house price prediction. Depending on the specific requirements of your model and dataset, you may need to perform additional preprocessing steps, such as encoding categorical variables or handling outliers.

Building the project by performing different activities like feature engineering, model training

Certainly! After preprocessing, the next steps typically involve feature engineering, model training, and evaluation for house price prediction. Let's break down each step:

1. Feature Engineering:

Feature engineering involves creating new features or modifying existing ones to improve the performance of the model. Here are some common techniques:

a. *Handling Categorical Variables:*

- If your dataset has categorical variables, encode them using techniques like one-hot encoding or label encoding.

```
# Example one-hot encoding
X_train_encoded = pd.get_dummies(X_train_scaled,
columns=['categorical_column'])
X_test_encoded = pd.get_dummies(X_test_scaled, columns=['categorical_column'])
```

b. *Creating New Features:*

- Extract relevant information from existing features or create new features that might be useful for the model.

```
# Example: Combine square footage of different areas
X_train_encoded['total_square_footage'] = X_train_encoded['living_area'] +
X_train_encoded['garage_area']
X_test_encoded['total_square_footage'] = X_test_encoded['living_area'] +
X_test_encoded['garage_area']
```

2. Model Training:

Choose a regression model suitable for house price prediction and train it using the training dataset.

a. Selecting a Model:

- Choose a regression algorithm like Linear Regression, Random Forest Regression, or Gradient Boosting Regression. Import the necessary library and initialize the model.

```
from sklearn.linear_model import LinearRegression

model = LinearRegression()
```

b. Training the Model:

- Fit the model to the training data.

```
model.fit(X_train_encoded, y_train)
```

3. Evaluation:

Evaluate the model's performance using the test dataset.

a. Predictions:

- Use the trained model to make predictions on the test data.

```
y_pred = model.predict(X_test_encoded)
```

b. Evaluation Metrics:

- Use appropriate evaluation metrics for regression, such as Mean Absolute Error (MAE), Mean Squared Error (MSE), or R-squared.

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'MAE: {mae}')
print(f'MSE: {mse}')
print(f'R-squared: {r2}')
```

These metrics will give you an idea of how well your model is performing. You can then fine-tune your model, try different algorithms, or explore more advanced techniques based on the evaluation results.

Remember that the process of feature engineering, model training, and evaluation is often iterative. You may need to go back and adjust parameters, try different features, or explore more advanced techniques to improve your model's performance.

Done by

N.Manikandan