* 1 [00:40] Following the devastating impact of the covid-19 pandemic, as people are increasingly vaccinated, the economy is looking to get back on track. One of the leading indicators of economic recovery as things start to return to normal is the price of housing, representing the overall housing market demand relative to supply. As market conditions change, how can you know if you’re getting a good deal on a house, or if you’re paying too much? In our project, we were interested to see what kind of factors, if any, can be used to reliably predict housing prices and to see how our predicted prices compare to the asking price of a “for sale” home.  
  [total 00:40]
* 2 [02:15] The very first step in building a predictive model for housing prices was finding an appropriate data set that would not only provide information on the sale prices of homes, but also information about the houses themselves, and this proved to have it’s own inherent challenges.   
  Our first thought was that there are already real estate websites readily available that are used by lots of people when they’re looking for houses, often seeking particular features, like number of bedrooms and baths, and with certain price ranges in mind. Sites like Zillow and Trulia seemed like they would have a wealth of information available, but unfortunately, since that is their bread and butter, they do not make this data readily available to the general public.   
  In an effort to still make this idea work, we started working on writing a program that would scrape the requisite data from Zillow. The initial data available to us through scraping included things like number of bedrooms and bathrooms, square footage, address, and asking price, but one of the potential downfalls with scraping is the inconsistency of data available from property to property. Another concern was that while most of the houses listed the asking price, they may not show the sale price of the house once it’s sold, and as we know, the value of a thing is determined by what someone will pay for it. The benefit of scraping, on the other hand, is that we would be able to choose data from any location and it would be up to date with the current housing market.   
  While still pursuing the web scraping plan, we continued to look at other potential options. After some searching, we came across a Kaggle competition titled “*House Prices – Advanced Regression Techniques”* to *“Predict sales prices and practice feature engineering, RFs, and gradient boosting.”* This was a competition designed for those with some Python and R programming experience to work on their data preparation and modeling techniques. The benefit to using this data set is that it contained a considerable amount more information and it was already prepared.   
  [total 02:55]
* 3 [01:10] The data set provided for the competition could be considered an updated version of the classic Boston Housing data set. This data was compiled in 2011 from residential housing data in Ames, Iowa between the years of 2006 and 2010. The entire data set consists of 2,930 rows containing columns with 79 explanatory variables, an observation Id number, and the actual sale price of the home. This data was split roughly in half, with 1,460 rows acting as the training set and the remainder as the test set, from which the sale price target variable was removed.   
  The variables included in the data set consisted of 47 Categorical variables, split about evenly between nominal and ordinal variables, 29 Numeric variables, both continuous (such as square feet of the living space) and discrete (such as number of bedrooms), as well as 5 datetime variables (such as year built or sold).

Hypothesis? / 80% accuracy?

[01:20] Once we loaded the .CSV files into dataframes in Python, we wanted to get an initial read on the data, so we looked at things like the shape of the dataframes and the list of column names. We computed summary statistics for each column in order to see things like counts, averages, standard deviation, minimum and maximum values, and quartiles. We also looked at histograms of some of the variables in order to get an idea of the shape of their distributions.   
Despite the fact that the data was already in a pretty good format, there were still some odd and ends that needed to be cleaned up. To start with, we combined the Train and Test data sets into one large dataframe. This provided us an opportunity to catch and fix any little oddities in the data that may appear infrequently. Say, for example, a variable contained a single Null value and it was only found in the Test data set. If we didn’t account for that Null value because we were just looking at the Training data set, then it might not get appropriately imputed during the data preparation stage and it could cause an error later when attempting to predict Sale Prices on the Test data, or ultimately during model deployment on actual data if this were a real-world project.  
[total 05:25]

* 4 [1:45] On the topic of null values, our data set contained quite a few of them that needed to be imputed. For many of our categorical variables, the Null value actually represented the absence of something, like “no basement” or “no garage”. These were easy to deal with because the fact that they were absent was a piece of relevant information in and of itself, since a house with no garage could reasonably have a negative impact on the value of a home. These Null values were simply imputed with a new category label of ‘None’. Related to this, were variables like Basement square footage, which had been given a Null value in the absence of a basement. Null values for these variables were imputed with a ‘0’, which is a more accurate representation of the actual square footage of the basement. This took care of the vast majority of the Null values, leaving just a handful that we imputed with the mean value, if numerical, or the mode for categorical variables.   
  For the datetime variables, we would typically look into creating new features such as the day of the week or week of the year from a given timestamp. In this data, the datetime variables were already separated into individual months or years, so we didn’t have to do much with them.   
  Once we had the data cleaning sorted out, we separated the Test data from the Training data again. This was relatively straightforward because we had added ‘train’ and ‘test’ keys to the dataframe when we combined them, so we were able to filter by these keys when separating them back out.   
  [total 07:10]
* 5 [00:55] As part of our data understanding, we created scatter plots of our numerical variables to compare against the target SalePrice variable. This allowed us to get an early read on potential trends in our data and one of the best relationships with SalePrice that we noticed was with the GrLivArea variable, which is the Above Ground Living Area, given in square feet. So, basically the size of the house. Which makes sense, because bigger houses tend to cost more.  
  In addition to these scatter plots, we also generated box plots of our categorical variables to compare their category distributions to the SalePrice target and to see which of the categories, if any, have an obvious impact on sale price relative to the others within the variable.  
  [total 08:05]
* 6 [1:30] In continuing our data preparation, we had initially encoded all of our categorical variables into dummy variables, which left us with over 300 columns. While dummy variables are required for some modeling, this did leave us with a high degree of dimensionality. We later realized that this was not the best approach for some of the categorical variables, because about half of them were actually ordinal variables, meaning that the categories have an inherent order to them, increasing in a kind of step-wise manner. An example of this is the quality of the exterior material on the house, which ranges from Poor to Fair then Average, Good, and topping out at Excellent. Because of this, instead of breaking these categories into individual dummy variables, we used scikit-learn’s OrdinalEncoder function to convert them to a scale of values from low to high. In the case of the exterior material example, Poor was converted to 1, Fair to 2, and so on, all the way up to Excellent which was given a 5. For categorical variables containing a ‘None’ category, the ‘None’ was converted to a ‘0’ value. Using OrdinalEncoder allowed us to reduce the dimensionality of our data, while also retaining the relative ranking information that would have been lost when creating dummy variables. Dummy variables were still created from the remaining nominal categorical variables.  
  [total 09:35]
* 7 [01:35] Our last data preparation step prior to modeling was Feature Selection. Initially, we created a correlation matrix of our data set and used the variable correlations with our SalePrice target to generate a data subset. This data feature subset consisted of variables that had a correlation with SalePrice that was greater than 0.5. While the exact correlation cutoff is somewhat arbitrary, this provided us with a dataframe consisting of 16 columns, which is a reasonable start relative to the approximately 80 columns that we started with and the more than 200 that we had after creating dummy variables.  
  We later included several other feature selection methods using kind of a shotgun approach where we selected the top variables based on high F-statistic scores, high LightGBM (or Gradient Boosting Machine) value, high Logistic Regression value, and variables with high Mutual Information values. Having ranked the variables using these different methods, we selected the top 20 from each category and created data feature subsets of each to try out in our modeling. In addition to these, we created one more subset of the top 20 based on an overall combined rating from all of the feature selection methods.  
  [total 11:10]
* 8 [00:20] Now that we had our data cleaned and prepared, our next step was to load the various dataframes into R Studio in order to begin modeling. Before creating the additional feature sets, we compared both linear regression and decision tree models using the highly correlated data and, while the decision tree model proved better by most metrics, we preferred the linear regression model due to the continuous nature of the target variable.   
  [total 11:30]
* 9 [01:40] Following the feature selection and creation of multiple data subsets, we created linear regression models using the full data sets (one of which included the ordinal encoded data and the other which consisted of the ordinal encoded data as well as the addition of dummy variables) and then we modeled each of the feature data subsets as well.   
  The two full data sets performed nearly identically and were rated the highest in a comparison of model performance indices, but wanting to reduce the number of features required and not use the entire data set, the next best performing model used the high F-statistic data subset and had an R squared of 0.811. This was followed closely by the Overall highly rated subset, which had an R squared of 0.806. These two models performed pretty closely on all of the model performance indices. Several of the most significant contributing variables to the F-statistic model were based around various Quality aspects of the home as well as garage car capacity and being located in one of a handful of specific neighborhoods. Oddly, being near a park or greenspace actually had a negative impact on SalePrice.  
  [total 13:10]
* 10 [01:20] Another modeling technique we tried was Boosting Trees using Gradient Boosted Modeling (GBM). This method creates thousands of shallow decision trees, but at each split it only chooses between a random subset of the available features. Through this, we were able to determine the relative importance of variables used and just a few of them made up the majority of influence on the model: “Overall Quality” accounting for roughly a third of the influence, “Square Footage of Above Ground Living Area” making up about a quarter, and “Garage Area” having about half the influence of Living Area. Looking at these variables a little more closely, it would appear that there is quite a jump in price for housing as the quality approaches ratings of 9 or 10. The various Area variables show a roughly linear increase up to a point. Somewhat interesting was the “Fireplace Quality” variable, which had a bit of a jump from no fireplace to a low, poor quality rating, then another price jump when moving up from poor quality to Fair, but then only minor increases as quality improved through Average and Good ratings, indicating that perhaps the Fair, Average, and Good ratings could be binned together for modeling purposes. Oddly there was a decrease in price for Excellent quality, which would require further looking into.  
  [total 14:30]
* 11 [01:15] Finally, we tried out the SuperLearner package for ensemble modeling. This allowed us to compare multiple models simultaneously and it would automatically weight the models used in the ensemble based on how much they were contributing to the predictive power of the ensemble. We initially ran it with 25 potential models and removed any that were weighted with a coefficient of zero, indicating that they were not contributing to the ensemble. This was repeated three times to weed out the less useful models. Additionally, one model was removed after the third pass because the risk factor, or error the algorithm produced, was an order of magnitude higher than for the other models. This narrowed it down to running just four models in the ensemble, with only three being weighted above zero. These three are the extraTrees model, the speedlm model, and the generalized boosted regression model. We ran cross-validation with these and the ensemble itself showed the lowest overall risk, compared to each of the individual models. Notice the highest risk model was weighted zero and not included in the ensemble.   
  [total 15:45]
* 12 [01:15] We put a lot of work into this project, but as with most projects, there’s never enough time to accomplish everything you want, and trade-offs need to be made. Some of the challenges we faced during this project included selection of features and how we encode those features for use in modeling, as well as which models are the best fit with our data for prediction purposes. We also struggled as a group with contemporaneous communication because all of our group members are located in different time zones around the world.   
  Additional work that could be done on this project includes comparing the different feature subsets with different models to determine which has the most predictive value of our SalePrice target. We didn’t do much with outliers because there weren’t any significant outliers, but it might be useful to look at more closely to see if any additional improvements could be made. Also, normalization or transformation of variables with skewed distributions may show improvements in some models. Lastly, model deployment would be a critical step in the project implementation.  
  [total 17:00]