HOUSE PRICING PRODECTOR IN KING COUNTY, US.

PARTNER NAME:

Ahmad Amireh

Diala Abedalqader

Khansaa Mahmoud

Table of contents

1. Introduction

Given the dataset that records the houses price in King County Washington, US. The house prices are recorded along with some other features like number of bedrooms, number of bathrooms, area of the house etc.

The Goal is to use this data to explore the factors on which the price depends and build a model which will be able to predict the price of the house based on the related features.

2. Importing and understanding our data

We importing our data set to our environment (python).

Description of the data: -

We can see after exploring our data contains 21,613 houses and 21 columns (20 features and one target).

- id - Unique ID for each home sold
- date - Date of the home sale
- price - Price of each home sold
- bedrooms - Number of bedrooms
- bathrooms - Number of bathrooms, where .5 accounts for a room with a toilet but no shower
- sqft_living - square footage of the apartments interior living space
- sqft_lot - square footage of the land space
- floors - Number of floors
- waterfront - A dummy variable for whether the apartment was overlooking the waterfront or not 1's represents a waterfront property, 0's represents a non-waterfront property
- view - An index from 0 to 4 of how good the view of the property was, 0 - lowest, 4 - highest
- condition - An index from 1 to 5 on the condition of the apartment, 1 - lowest, 4 - highest
- grade - An index from 1 to 13, where 1-3 falls short of building construction and design, 7 has an average level of construction and design, and 11-13 have a high-quality level of construction and design.
- sqft_above - The square footage of the interior housing space that is above ground level
- sqft_basement - The square footage of the interior housing space that is below ground level
- yr_built - The year the house was initially built
- yr_renovated - The year of the house's last renovation

- zipcode - What zipcode area the house is in
- lat - Lattitude
- long - Longitude
- sqft_living15 - The square footage of interior housing living space for the nearest 15 neighbors
- sqft_lot15 - The square footage of the land lots of the nearest 15 neighbors

3. Cleaning and validate the dataset

1) data types: - convert the datatype because it is important because every data type has its own method that can help in our analysis. Also, having a correct data type it is crucial in data visualization.

   Examples:

   a. convert the date timestamp to the standard format
   b. convert “condition","grade","view","waterfront" from int to category because they have a finite set of possible values
2) dealing with missing values: there is no missing values in our dataset
3) dealing with duplicated values: there is some houses that sold more than one in 2014 and 2015 but its only 176 out of 21,613 so we drop it.
4) validate our data: sqft_living = sqft_basement + sqft_above so we check if all rows are correct and it appears that our data is trustworthy
5) checking inconsistency in categorical/discrete columns: checking that all categorical/discrete columns have the same range as a description and has a valuable meaning.

   Examples: after checking bedrooms we can see that there are houses with no bedrooms which doesn’t make sense wo we drop them. Also, there is one house with 33 bedrooms which also doesn’t make sense and it consider outlier so we drop it
6) deleting features that doesn’t have any impact (ID column)

NOTE: EDA hasn’t been required to document it in this project so we will proceed with feature extraction

4. Feature Extraction

1. convert zip code column to city name to get more valuable information in our dataset using USZIP CODE LIBRARY. also deleting (lat and long columns until now)
2. convert the yr_renovated column to is_renovated column because most of the houses hasn't been renovated and the goal is to only know that the house is_renovated or not.
3. from the date we extract the year and month of purchase

5. Feature Selection

We use Backward elimination method, which is is a feature selection method used in machine learning. It involves starting with a full set of features and iteratively removing the least significant features one by one until only the most significant features remain

choose 0.05 as significance level.

before performing it, we need to prepare our data:

a) City column must be converted to numerical represent so one hot encoding has been performed on this column using (pd.get_dummies) and droping the first city column to not fall into Dummy variable Trap.

b) split feature(X) from target (y) and convert both to array using ". values" method.

c) The library we use to perform Backward elimination (statsmodels) doesn't have b0(intercept) by default so we need to create a add a column at the first of the feature array

d) We create the model using statsmodels and we get the summary from it, which has the p_value for all features that indicate the relationship between this feature and the target. we start by deleting the higher p_value that exceed our significance level and we keep doing this until all features have p_value less than 0.05 which indicate strong relationship with target.

e) Only three columns are not important. so df_after_extract_features which is a dataframe that has all features that we want.

['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'waterfront', 'view', 'condition', 'grade', 'sqft_above', 'sqft_basement', 'yr_built', 'sqft_living15', 'sqft_lot15', 'is_renovated', 'year', 'month', 'Bellevue', 'Black Diamond', 'Bothell', 'Carnation', 'Duvall', 'Fall City', 'Federal Way', 'Issaquah', 'Kenmore', 'Kirkland', 'Maple Valley', 'Medina', 'Mercer Island', 'North Bend', 'Redmond', 'Renton', 'Sammamish', 'Seattle', 'Snoqualmie', 'Woodinville'].

Finally, merge df_after_extract_features with our target variable to have df_new, which is the dataframe will enter our model.

NOTE: we do the same procedure but when we consider the lat and lang in our features.

And named df_lat. (all the features are considered as important when we add lat and long)

6. MODELING (MACHINE LEARNING)

1. Regression_models ()

This class for random forest and gradient boosting algorithms, which contains functions that help automating any data to choose the best model.

This **class has an input variable** which is:

- data (panda DataFrame): Our data after analysis, feature selection and extraction
- target (Pandas Series, NumPy array): the target variable that we want to predict which is in our case price
- algorithm (class in sklearn): Random Forest or Gradient boosting using scikti-learn to call one of it

function include this class:

1- **model function**:

Description:

this function splitting our features (X) from our target (y) and also splitting our data to train and test data to get reliable result on unseen data.

Input: only get its input from class.

Output (tuple):

- X_train: which are the features on the training data.
- Y_ train: which is the target variable on the train data
- X_test: which are the features on the test data
- Y_test: which is the target variable on the test data

2- **Train function**:

Description:

training the model to get the coefficients by fitting the X_Train and y_train on it which is now the return values from model function

Input: get its input from our class.

Output (model): training our model in X_train and y_train

3- **Accuracy function**:

Description:

crating folds for validation data, then calculate the accuracy for cross validation of data, training, and testing the data

Input:

- number_of_fold (int): number of folds want to split our train data to create CV.
- get its input from our class variable, train function and model function in our class

Output (tuple):

- return accuracy for cross validation of data, training, and testing the data.

4- **enhance_rf_model**:

Description:

Hyperparameter tuning for the random forest model, to get best hyperparameter.using RandomizedSearchCV in sklearn which takes arguments (our model ,a dict of params and its possible range , cross_validation and n_iters which is the combination that it will take .

Input:

- n_estimators (list/array/int) = The number of trees in the forest., default value =100
- max_depth (int /array/list) = The maximum depth of the tree, default value=8
- min_sample_leaf (int /array/list) = The minimum number of samples required to split an internal node, default value =3
- number_of_fold (int)= number of k fold, default value=5

Output (dictionary):

return best hyperparameters after grid search

NOTE: other parameters only to show the default value.

5- enhance_boost_model:

Description:

Hyperparameter tuning for the gradient boosting model, to get best trees number (n_estimators) at a certain learning rate which is in our case 0.2. using RandomizedSearchCV in sklearn.

input:

- n_estimators(int) = The number of boosting stages to perform.
- number_of_fold(int)= number of k fold, default value=5

output: return best n_estimators at a certain learning rate after grid search

6- enhance_boost_model_second

Description:

Hyperparameter tuning for the gradient boosting model at a certain learning rate and n_estimators, to get best hyperparameter, related to the trees using RandomizedSearchCV in sklearn

Input:

- max_depth: -The maximum depth of the tree, default value=8
- min_sample_leaf: The minimum number of samples required to split an internal node, default value =10

output:

return best max_depth and min_sample_leaf at a certain learning rate and n_estimators

NOTE: other arguments only to show the default value.

7- feature selection:

Description:

most important features, depending on how much tree nodes use a particular feature (weighted average) to reduce impurity by using feature_importances_ in sklearn.

input:

get its input from our class.

output (plot):

bar plot which is indicate the most important feature.

8-learn curve:

Description:

plot curves show us how well the model is performing as the data grows.

which contains two curves:

- Training curve: The curve calculated from the training data; used to inform how well a model is learning.
- Validation curve: The curve calculated from the validation data; used to inform of how well the model is generalizing to unseen instances.

Input (array/list):

training size: the absolute numbers of training examples that will be used to generate the learning curve; the values we are using are completely random. (x-axis of our curve)

Output (plot):

two lines plot:

a) train size vs train error

b) train size vs test error

9. price_var.

Description:

Finding the residual which between the actual and perdition price to see how its distribution looks links.

Input: from class.

Output: plot distribution for residual

10 RMSE function

Description:

Calculate the root mean square error

Input : from class

Output: float number for root mean square error

2. **Feature_selection ()**

This class for selecting the most important features using random forest and lasso in order to get high performance and minimize the training time.

This **class has an input variable** which is:

- data (panda DataFrame): Our dataset
- target (Series / NumPy array): the target variable that we want to predict which is in our case price

function include the class:

1)Random_forest_method:

Description:

most important features, depending on random forest selection.

- splitting our dataset into training and testing set
- inistantiate RandomForest class and train it
- create a series with our purity using feature_importance and names of features and sort it with descending order
- filter our data to create data frame with only features with high than 0.05

Input:

get its input from our class

Output (tuple):

- DataFrame that contains only the most importance features (index 0)

- bar plot to show visually the most important feature (index 1)

2- **Lasso function**:

Description:

most important features, depending on lasso coefficients (ignoring the coefficient which shrink to zero)

- splitting our dataset into training and testing set
- create a pipeline that contains Lasso class from sklearn and Standard Scaler in order to instantiate the model and doing the preprocessing at the same time
- using GridSearchCV in order to pick the best alpha
- find the coefficients using the best model(best_estimator) that result from gridSearch model and taking the absolute value from it.
- Create a Series with coefficient of the model and name of features
- filter this data to take only coefficient with higher than 0 (only important feature)

input:

k_fold (int): number of folds want to split our train data to create CV.

Output (tuple):

- DataFrame which contains the most important feature (index 0)
- bar plot to show visually the most important feature (index 1)

7. Saving the best trained model

After creating two models and doing enhancement on it, it has been observed that Gradiant Bosssting has accuracy (0.9) higher than RandomForest (0.86) so it will be taken.

Then, we save our best model using pickle library in order to load the model in a user notebook.

8. Creating User friendly interface

In this part we show the steps in user notebook.

1. The file that has the best model has been loaded into user notebook.
2. After this the function “full**_input_data** “has been created:

   1. **Description**:

      it will take any input for information that related to the house that user want and give him the price of the house.

      - create data frame which contains input that doesn’t related to one hot encoding column
      - crate data frame for the one hot encoding column (city)
      - concatenate first step with the second step to give us the whole data frame that will enter to our model (OUR X_TEST)

      input:

      - data_before_dm (DataFrame): our data before performing one hot encoding
      - city_name (string): the city of the house. default = “Seattle”
      - input_data (list): input of features that doesn’t related to one hot encoder

output (1d array):

return our prediction (y_pred)

3. creating web page using streamlit framework.

   The function main has been created:

   Description:

   creating interactive page, for the user to enter any input.

   - number_input for feature doesn’t relate to one hot encoding
   - select_box to the city column (one hot encoding has been performed on it)
   - button: for our target variable and connect it with our “full input data “, so when user enter the button, the prediction will come.

   input: None

   output: web page (user interface)

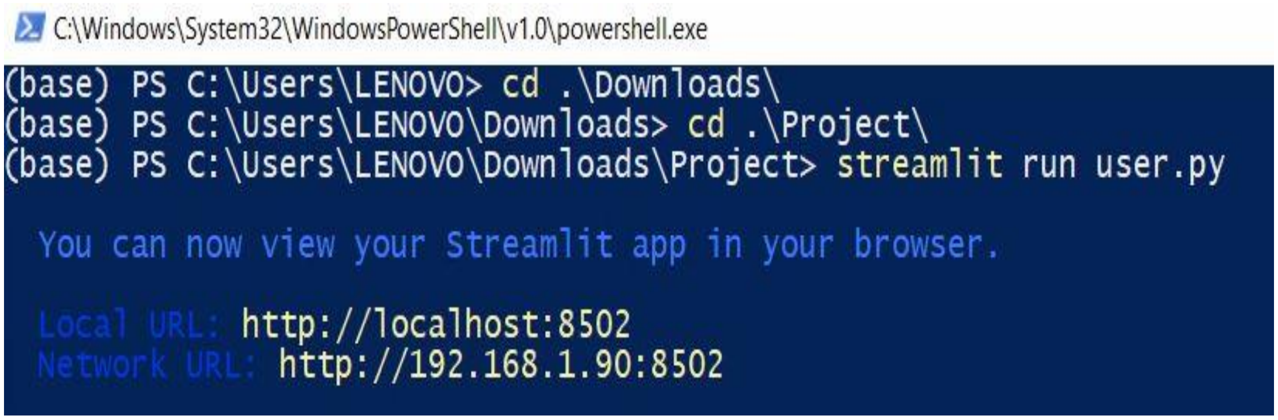
To display the GUI run the python file from the terminal to display your local host webpage by

giving it the path of your file as shown in figure 1. And it will automatically redirect you to the

web page as shown in figure 2.

Steps : 1- go to anaconda shell

2- write on it as figure shown below , which is the directory of the file and run streamlit name of the file after this press enter .



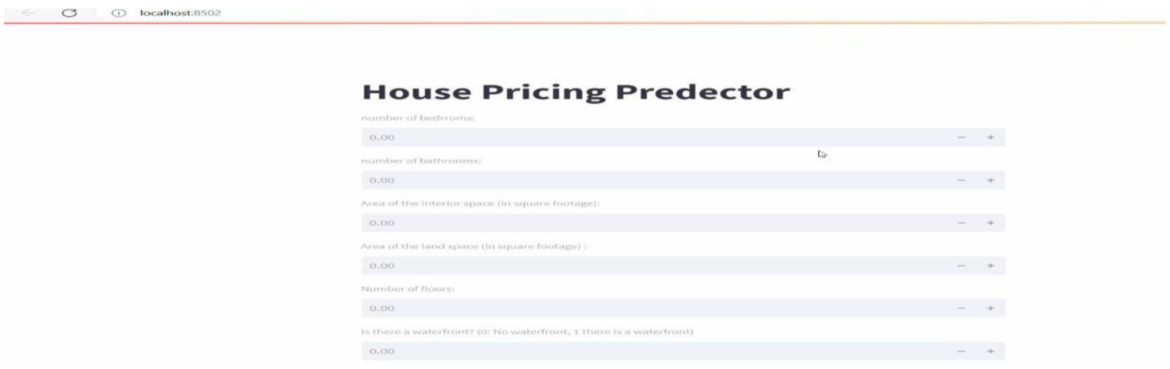
```
C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
(base) PS C:\Users\LENOVO> cd .\Downloads\
(base) PS C:\Users\LENOVO\Downloads> cd .\Project\
(base) PS C:\Users\LENOVO\Downloads\Project> streamlit run user.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8502
  Network URL: http://192.168.1.90:8502
```

Figure (1)

3- It will open the user interface directly which is shown in figure 2 :



localhost:8502

House Pricing Predector

number of bedrroms:
0.00

number of bathrooms:
0.00

Area of the interior space (in square footage):
0.00

Area of the land space (in square footage) :
0.00

Number of floors:
0.00

Is there a waterfront? (0: No waterfront, 1 there is a waterfront)
0.00

Figure(2)

Results

In this part we look in the results for the steps we through in this project

Result 1:

From feature extraction the result is: try as much as possible to extract the most valuable info from our data.

Result 2:

From feature selection the result is: having two dataframe resulting after backward elimination:

- df_new (without latitude and longitude)
- df_lat (with latitude and longitude)

So, we need to test them in our model to see which one will give us the higher score.

Result 3:

from **Regression_model class the result is:**

this class can create RandomForest and GradiantBoosting Regressor for any data, calculate the accuracy, enhance models and checking it generalize our data or not by using learning curve.

Result 4:

from **Feature_selection class:**

create this class to show if it gives us a different result than back word elimination so maybe if there any different it will enhance the model but there **is no different in our case**.

Result 5:

This result to show that GradiantBoostingRegressor using df_lat has the higher score than the rest of models. also, it has a good Generalization Error (overfitting and underfitting issue has been solved using hypermeter tuning) and making user interface for it.