Kings County house prediction model

Table of Contents

Introduction:

This machine learning model is designed to predict the sale prices of homes in Kings County. It takes a variety of features into account, such as the size of the home, the number of bedrooms and bathrooms, and the location, to make its predictions.

1- Purpose of the predictive model

Problem description : The problem that this model tries to solve is predicting the sale prices of homes in Kings County. By analyzing trends in the housing market and using a variety of relevant features, the model can provide valuable insights to homeowners, real estate agents, and investors.

By accurately predicting the sale prices of homes, the model can help individuals make informed decisions about buying or selling property in Kings County. It can also help investors identify potentially undervalued or overvalued properties, and inform the development of real estate investment strategies.

Overall, the model aims to provide a useful tool for understanding and navigating the housing market in Kings County.

Model behavior : The behavior that this model is trying to predict is the sale price of homes in Kings County. This behavior will be defined and measured in terms of monetary value, specifically in U.S. dollars.

To define and measure the behavior, the model will be trained on a dataset of past home sales in Kings County. The dataset will include various features for each home, such as the size, number of bedrooms and bathrooms, and location, as well as the sale price.

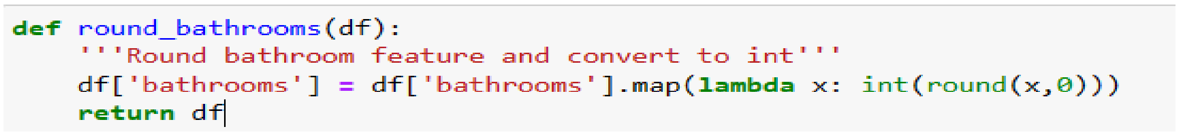
The model will use this data to learn the relationships between the different features and the sale price, and will be able to make predictions on the sale price of new homes based on those relationships. The accuracy of the model's predictions will be evaluated using metrics such as mean absolute error and root mean squared error, which measure the average magnitude of the error made by the model.

Overall, the behavior that the model is trying to predict is the sale price of homes in Kings County, and this behavior will be defined and measured in terms of monetary value.

Data source : the data was provided and used as predictors in this model was a data set on past home sales in Kings County, this data would include information on the features of each home (e.g., size, number of bedrooms and bathrooms, location) as well as the sale price. This data could be used to train and validate the model, and would help the model learn the relationships between the different features and the sale price.

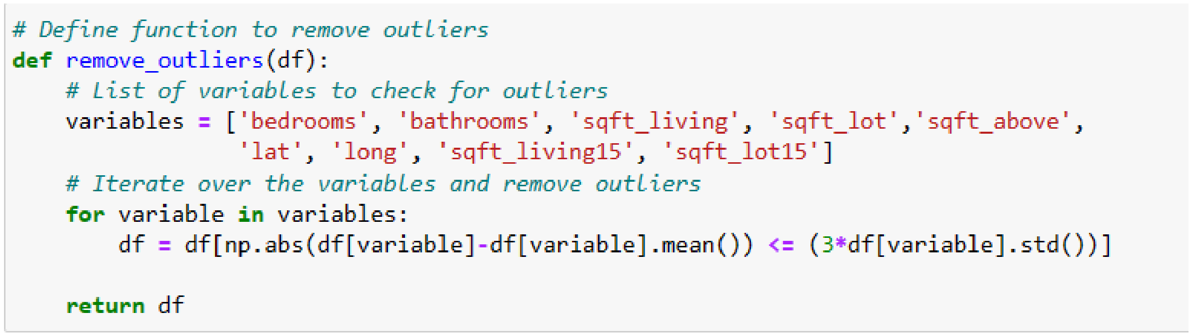
2- Model building:

a- **Data preparation:** the following customized function was used to prepare & clean the data before model building.



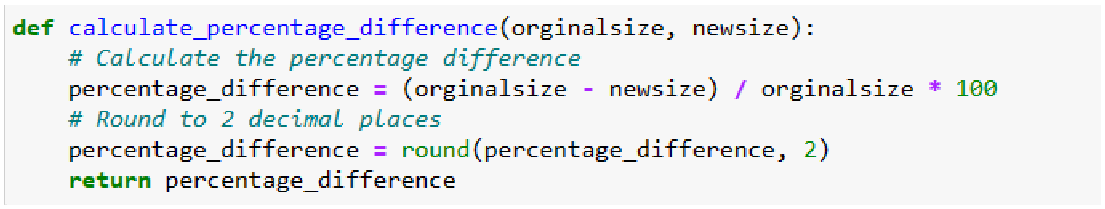
```
def round_bathrooms(df):
    '''Round bathroom feature and convert to int'''
    df['bathrooms'] = df['bathrooms'].map(lambda x: int(round(x,0)))
    return df
```

This function takes in a data frame and rounds the values in the 'bathrooms' column to the nearest integer. It then converts these rounded values to integers. The modified data frame is then returned.



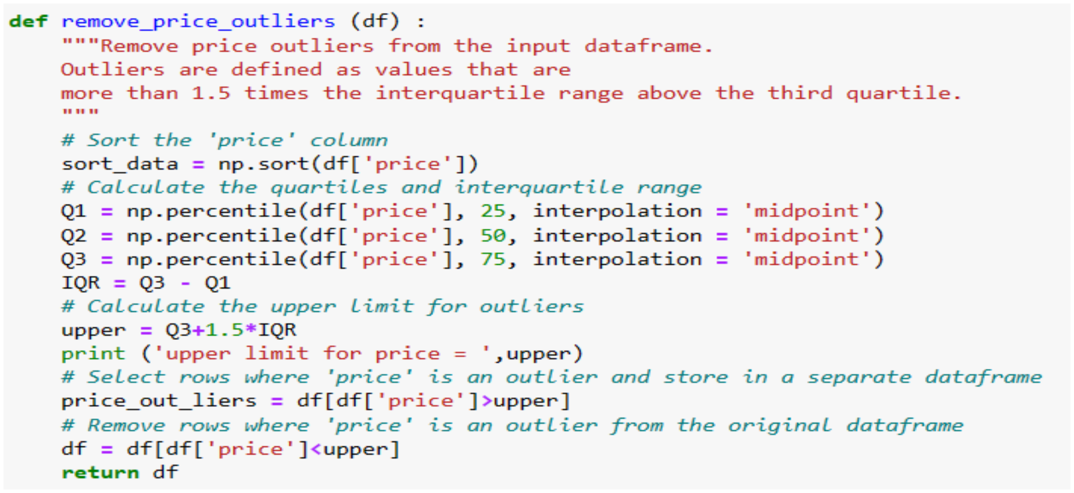
```
# Define function to remove outliers
def remove_outliers(df):
    # List of variables to check for outliers
    variables = ['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot','sqft_above',
                 'lat', 'long', 'sqft_living15', 'sqft_lot15']
    # Iterate over the variables and remove outliers
    for variable in variables:
        df = df[np.abs(df[variable]-df[variable].mean()) <= (3*df[variable].std())]

    return df
```

This function takes in a data frame and removes any outliers from certain columns in the data frame. An outlier is a value that is significantly different from the other values in a dataset and could potentially skew the results of an analysis. The function identifies and removes these values by keeping only the rows where the value in a column is within a certain range relative to the mean and standard deviation of the column. The modified data frame is then returned. This function can be useful for cleaning and pre-processing a dataset before using it to train a machine learning model.



```python
def calculate_percentage_difference(orginalsize, newsize):
    # Calculate the percentage difference
    percentage_difference = (orginalsize - newsize) / orginalsize * 100
    # Round to 2 decimal places
    percentage_difference = round(percentage_difference, 2)
    return percentage_difference
```

The function first calculates the percentage difference by subtracting new size of the data set from the original size, and dividing the result by the original size. The result is then multiplied by 100 to express it as a percentage.
The percentage difference is then rounded to two decimal places using the round () function. Finally, the function returns the percentage difference.
This function used for comparing the size of two quantities and expressing the difference as a percentage.



```python
def remove_price_outliers (df) :
    """Remove price outliers from the input dataframe.
    Outliers are defined as values that are
    more than 1.5 times the interquartile range above the third quartile.
    """
    # Sort the 'price' column
    sort_data = np.sort(df['price'])
    # Calculate the quartiles and interquartile range
    Q1 = np.percentile(df['price'], 25, interpolation = 'midpoint')
    Q2 = np.percentile(df['price'], 50, interpolation = 'midpoint')
    Q3 = np.percentile(df['price'], 75, interpolation = 'midpoint')
    IQR = Q3 - Q1
    # Calculate the upper limit for outliers
    upper = Q3+1.5*IQR
    print ('upper limit for price = ',upper)
    # Select rows where 'price' is an outlier and store in a separate dataframe
    price_out_liers = df[df['price']>upper]
    # Remove rows where 'price' is an outlier from the original dataframe
    df = df[df['price']<upper]
    return df
```

This is a function that removes outliers from the 'price' column of a data frame. An outlier is defined as a value that is more than 1.5 times the interquartile range above the third quartile of the 'price' column.
The function first sorts the 'price' column using the NumPy function np.sort(). It then calculates the quartiles and interquartile range of the 'price' column using the NumPy function np.percentile(). The upper limit for outliers is then calculated as the third quartile plus 1.5 times the interquartile range.

The function then selects the rows where the 'price' column is an outlier and stores them in a separate data frame. The original data frame is then modified to remove these rows. Finally, the modified data frame is returned by the function.

---

b- Data splitting: The data used for training and validation will be divided into two samples; a training sample and a test sample.
The training sample is a subset of the data that is used to fit the model. The model will learn the relationships between the different features and the sale price from this sample, and will use these relationships to make predictions.
The test sample is a subset of the data that is held out until the end of the model training process. It is used to get a final evaluation of the model's performance and ensure that it is not overfitting.

```
# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size = 0.1, random_state =0)
```

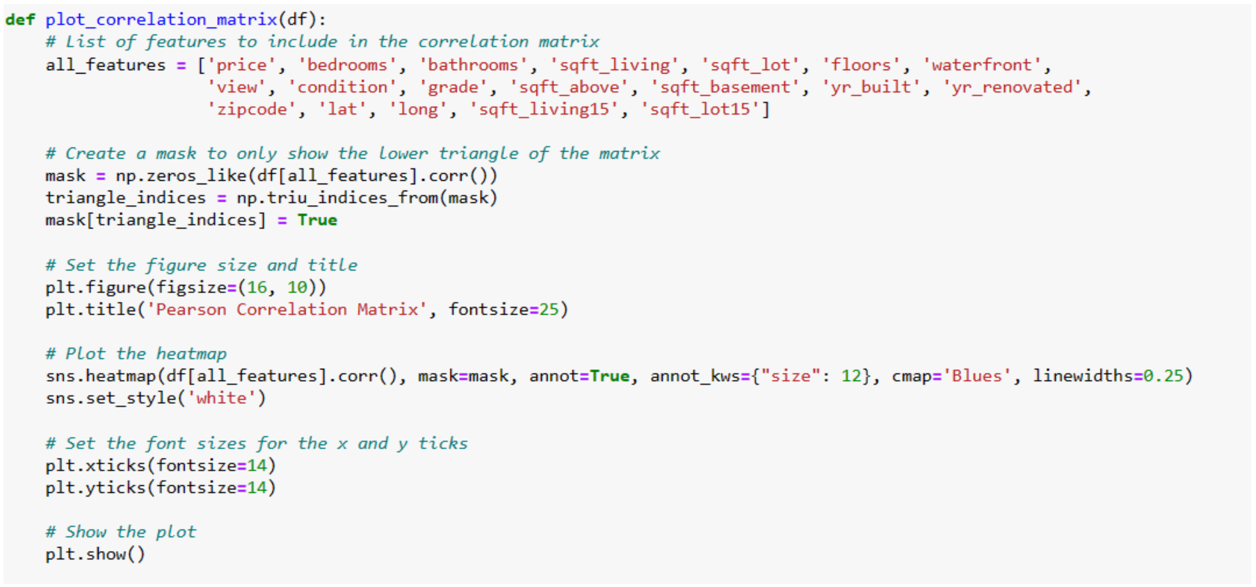
This code uses the train_test_split() function from the scikit-learn library to split a dataset into training and test sets. The function takes four arguments as input:

X: The input data, house price sales data set
y: The target variable, '' price ''
test_size: The proportion of the dataset to include in the test set. This value can be a float between 0 and 1, or an integer representing the number of samples to include in the test set.
random_state: The random seed to use when shuffling the data. This is an optional argument that can be used to ensure that the same split is obtained every time the function is run.

The function returns four objects:

X_train: The input data for the training set.
X_test: The input data for the test set.
y_train: The target variable for the training set.
y_test: The target variable for the test set.

c- **Features extraction:** Pearson correlation matrix is used to identify the features in the dataset that are most strongly correlated with the target variable (price).



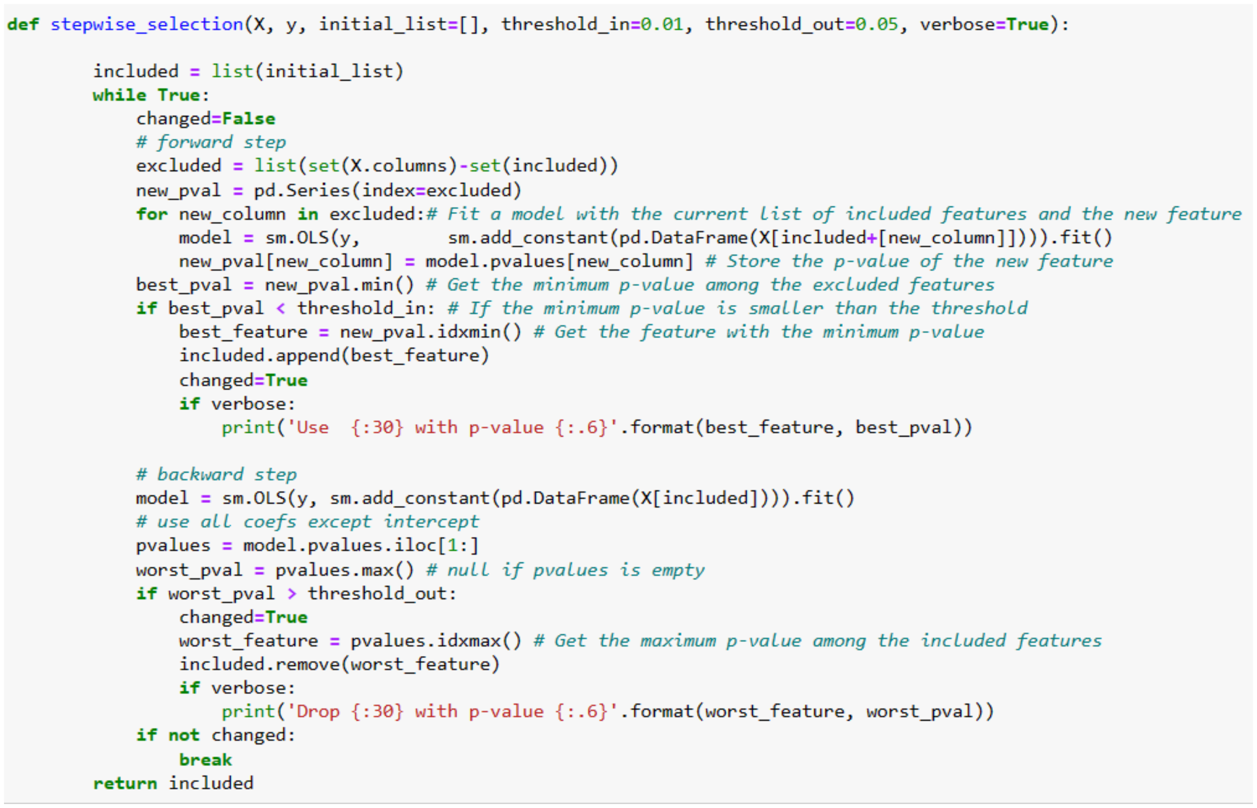
```
def plot_correlation_matrix(df):
    # List of features to include in the correlation matrix
    all_features = ['price', 'bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'waterfront',
                    'view', 'condition', 'grade', 'sqft_above', 'sqft_basement', 'yr_built', 'yr_renovated',
                    'zipcode', 'lat', 'long', 'sqft_living15', 'sqft_lot15']

    # Create a mask to only show the lower triangle of the matrix
    mask = np.zeros_like(df[all_features].corr())
    triangle_indices = np.triu_indices_from(mask)
    mask[triangle_indices] = True

    # Set the figure size and title
    plt.figure(figsize=(16, 10))
    plt.title('Pearson Correlation Matrix', fontsize=25)

    # Plot the heatmap
    sns.heatmap(df[all_features].corr(), mask=mask, annot=True, annot_kws={"size": 12}, cmap='Blues', linewidths=0.25)
    sns.set_style('white')

    # Set the font sizes for the x and y ticks
    plt.xticks(fontsize=14)
    plt.yticks(fontsize=14)

    # Show the plot
    plt.show()
```

The function takes a data frame as input and defines a list of features to include in the correlation matrix. It then creates a mask to show only the lower triangle of the matrix, as the upper triangle is a mirror image of the lower triangle.
The function then uses the seaborn library's heatmap() function to plot the correlation matrix, using the mask to show only the lower triangle. The function also sets the figure size and title, and sets the font sizes for the x and y ticks. Finally, the function shows the plot using the show () function.

Selected features for this prediction model:
1- No of bedrooms
2- No of bathrooms
3- Square-feet living area
4- Square-feet lot area
5- No of floors
6- House waterfront
7- No of times viewed
8- House condition
9- House grade
10- Square feet basement area
11- Year built
12- Year renovated

13- Lateral coordinate
14- Longitude coordinate

Stepwise function: It is often used in machine learning to improve the interpretability and performance of a model by reducing the number of features that are included in the model.
This function is performing stepwise feature selection on the dataset, and used for selecting a subset of features from a larger set that are most relevant to the target variable.



```python
def stepwise_selection(X, y, initial_list=[], threshold_in=0.01, threshold_out=0.05, verbose=True):

        included = list(initial_list)
        while True:
            changed=False
            # forward step
            excluded = list(set(X.columns)-set(included))
            new_pval = pd.Series(index=excluded)
            for new_column in excluded:# Fit a model with the current list of included features and the new feature
                model = sm.OLS(y,        sm.add_constant(pd.DataFrame(X[included+[new_column]]))).fit()
                new_pval[new_column] = model.pvalues[new_column] # Store the p-value of the new feature
            best_pval = new_pval.min() # Get the minimum p-value among the excluded features
            if best_pval < threshold_in: # If the minimum p-value is smaller than the threshold
                best_feature = new_pval.idxmin() # Get the feature with the minimum p-value
                included.append(best_feature)
                changed=True
                if verbose:
                    print('Use  {:30} with p-value {:.6}'.format(best_feature, best_pval))

            # backward step
            model = sm.OLS(y, sm.add_constant(pd.DataFrame(X[included]))).fit()
            # use all coefs except intercept
            pvalues = model.pvalues.iloc[1:]
            worst_pval = pvalues.max() # null if pvalues is empty
            if worst_pval > threshold_out:
                changed=True
                worst_feature = pvalues.idxmax() # Get the maximum p-value among the included features
                included.remove(worst_feature)
                if verbose:
                    print('Drop {:30} with p-value {:.6}'.format(worst_feature, worst_pval))
            if not changed:
                break
        return included
```

The function takes several arguments as input:

X: A data frame of features.
y: A vector of target values.

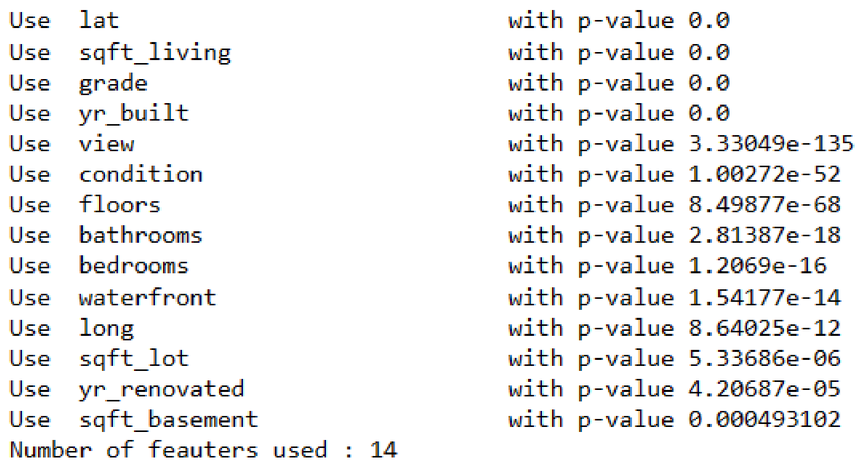
initial_list: A list of features to include in the model initially. This is an optional argument with a default value of an empty list.

threshold_in: The p-value threshold for adding a new feature to the model. This is an optional argument with a default value of 0.01.

threshold_out: The p-value threshold for removing a feature from the model. This is an optional argument with a default value of 0.05.

verbose: A Boolean value indicating whether to print the names of the features that are added or removed from the model. This is an optional argument with a default value of True.

The function performs stepwise selection by iteratively adding and removing features from the model until no more changes are made. The function uses the stats model's library to fit linear regression models and calculate p-values for each feature. If the p-value of a feature is below the threshold_in value, it is added to the model. If the p-value of a feature is above the threshold_out value, it is removed from the model. The function returns a list of the features that are included in the final model.

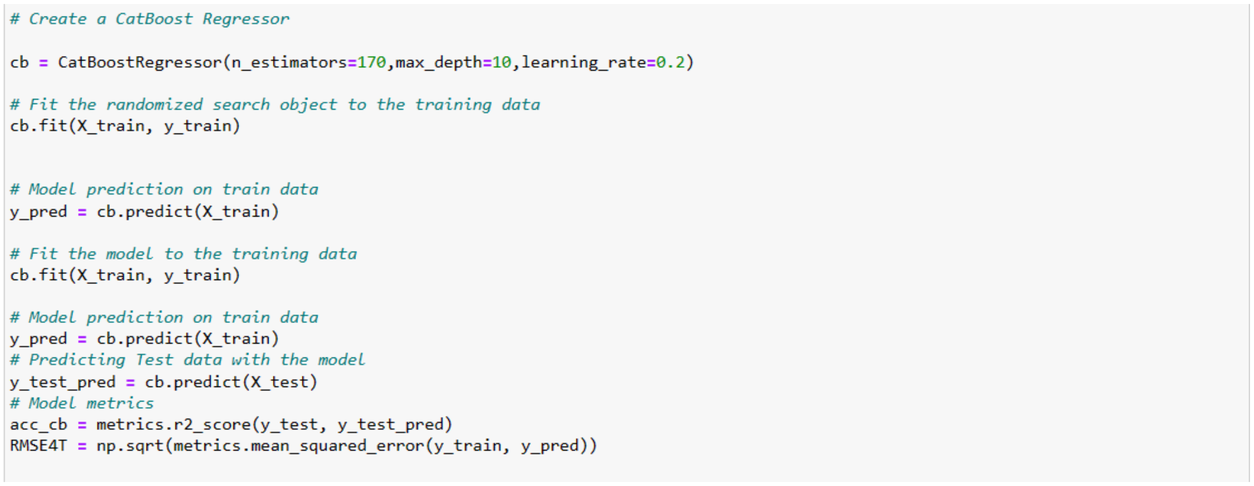


```
Use  lat                          with p-value 0.0
Use  sqft_living                  with p-value 0.0
Use  grade                        with p-value 0.0
Use  yr_built                     with p-value 0.0
Use  view                         with p-value 3.33049e-135
Use  condition                    with p-value 1.00272e-52
Use  floors                       with p-value 8.49877e-68
Use  bathrooms                    with p-value 2.81387e-18
Use  bedrooms                     with p-value 1.2069e-16
Use  waterfront                   with p-value 1.54177e-14
Use  long                         with p-value 8.64025e-12
Use  sqft_lot                     with p-value 5.33686e-06
Use  yr_renovated                 with p-value 4.20687e-05
Use  sqft_basement                with p-value 0.000493102
Number of feauters used : 14
```

d- **Choosing the appropriate model:** choosing the appropriate model will depend on the characteristics and requirements of the data and the exploratory data analysis for house prices data set, this will help to narrow down the list of potential models and ensure that the chosen model is appropriate for the task at the hand.
On the other hand, determine the type of model that is most appropriate for the data and the problem, there are many different types of models to choose from, including linear models and tree-based models.
After choosing the models needed to be tested it is a must to consider the model's performance and complexity and how is important to find a balance between performance and complexity.

Finally evaluating the model's performance using appropriate metrics, once a model has been selected, it is important to evaluate its performance on the dataset, this will help to determine the model's strengths and weaknesses and identify any areas for improvements.

CatBoost Regressor model:



```
# Create a CatBoost Regressor

cb = CatBoostRegressor(n_estimators=170,max_depth=10,learning_rate=0.2)

# Fit the randomized search object to the training data
cb.fit(X_train, y_train)


# Model prediction on train data
y_pred = cb.predict(X_train)

# Fit the model to the training data
cb.fit(X_train, y_train)

# Model prediction on train data
y_pred = cb.predict(X_train)
# Predicting Test data with the model
y_test_pred = cb.predict(X_test)
# Model metrics
acc_cb = metrics.r2_score(y_test, y_test_pred)
RMSE4T = np.sqrt(metrics.mean_squared_error(y_train, y_pred))
```

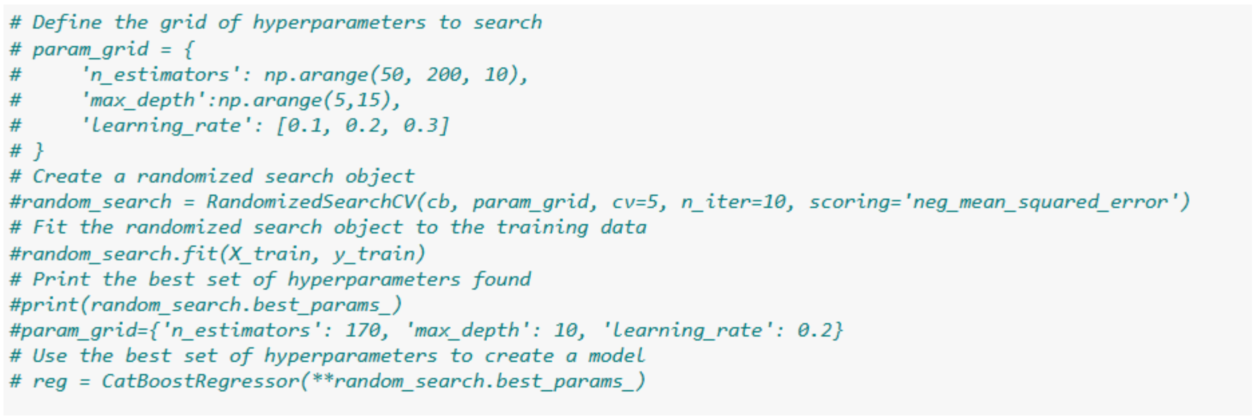
This code uses the CatBoostRegressor class from the CatBoost library to create a CatBoost model for regression. The model is trained using the fit method with the X_train and y_train data. The model's performance on the training data is evaluated using the predict method and the r2_score and mean_squared_error metrics from the metrics module. The model's performance on the test data is also evaluated using the predict method and the r2_score and mean_squared_error metrics.

The n_estimators parameter specifies the number of decision trees to use in the model, the max_depth parameter specifies the maximum depth of the trees, and the learning_rate parameter specifies the learning rate used in training the model. These parameters can be fine-tuned to improve the model's performance.

The RMSE4T variable stores the root mean squared error (RMSE) of the model's predictions on the training data. The RMSE is a measure of the difference between the predicted values and the true values. A lower

RMSE indicates a better fit. The acc_cb variable stores the model's accuracy on the test data, as measured by the R2 score. The R2 score is a measure of how well the model fits the data, with a score of 1 indicating a perfect fit.

Best Parameters to use - using Randomized Search:

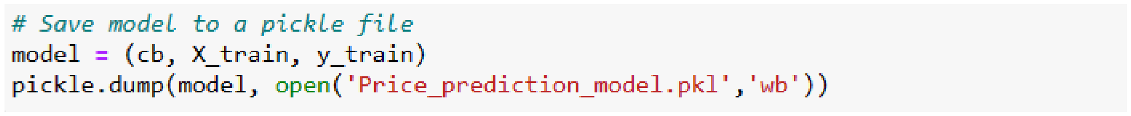


```
# Define the grid of hyperparameters to search
# param_grid = {
#     'n_estimators': np.arange(50, 200, 10),
#     'max_depth':np.arange(5,15),
#     'learning_rate': [0.1, 0.2, 0.3]
# }
# Create a randomized search object
#random_search = RandomizedSearchCV(cb, param_grid, cv=5, n_iter=10, scoring='neg_mean_squared_error')
# Fit the randomized search object to the training data
#random_search.fit(X_train, y_train)
# Print the best set of hyperparameters found
#print(random_search.best_params_)
#param_grid={'n_estimators': 170, 'max_depth': 10, 'learning_rate': 0.2}
# Use the best set of hyperparameters to create a model
# reg = CatBoostRegressor(**random_search.best_params_)
```

This code shows an alternative method for defining the hyperparameter grid and creating the CatBoost model using a fixed set of hyperparameters.
This code defines a grid of hyperparameters to search for the best combination of n_estimators, max_depth, and learning_rate values to use in the CatBoost model. The RandomizedSearchCV function from scikit-learn is then used to perform a random search over the defined hyperparameter grid. The search is performed using 5-fold cross-validation and 10 iterations. The neg_mean_squared_error scoring metric is used to evaluate the model's performance.

Once the random search has been completed, the best_params_ attribute of the random_search object is used to get the best set of hyperparameters found. These hyperparameters can then be used to create a new CatBoost model using the CatBoostRegressor class and the ** operator to unpack the dictionary of hyperparameters.

Model saving:



```
# Save model to a pickle file
model = (cb, X_train, y_train)
pickle.dump(model, open('Price_prediction_model.pkl','wb'))
```

This code saves a trained machine learning model to a file using the pickle library. The model is first created as a tuple containing the trained model object ('cb'), the training data ('X_train' and 'y_train'), and then passed to the pickle.dump() function along with the file name ('Price_prediction_model.pkl') and the write ('wb') mode.

The pickle library is a Python library that is used to serialize and deserialize Python objects, allowing them to be saved to and loaded from files. This is useful for storing trained models so that they can be used later without the need to retrain the model from scratch.

3- Model implementation:

The code is a script for creating a graphical user interface (GUI) that allows a user to input various details about a house and predict its price. The script defines two functions, open window and pridect_calc, which handle different aspects of the program.

```python
def open_window(result,sqft_liv):
    price_sqft=round(result/sqft_liv)
    result=round(result)
    layout = [[sg.Text('Price of the house: {} USD'.format(result), key="new")],
              [sg.Text('Price/Sq.Ft: {} USD'.format(price_sqft), key="new")],
              [sg.Button('ok', size=(10, 1), button_color=('white', '#3498db'), font=('Helvetica', 12))]]
    window = sg.Window("Second Window", layout, modal=True)
    choice = None
    while True:
        event, values = window.read()
        if event == "Exit" or event == sg.WIN_CLOSED:
            break
        elif event == 'ok':
            window.close()
            pridect_calc()
    window.close()
```

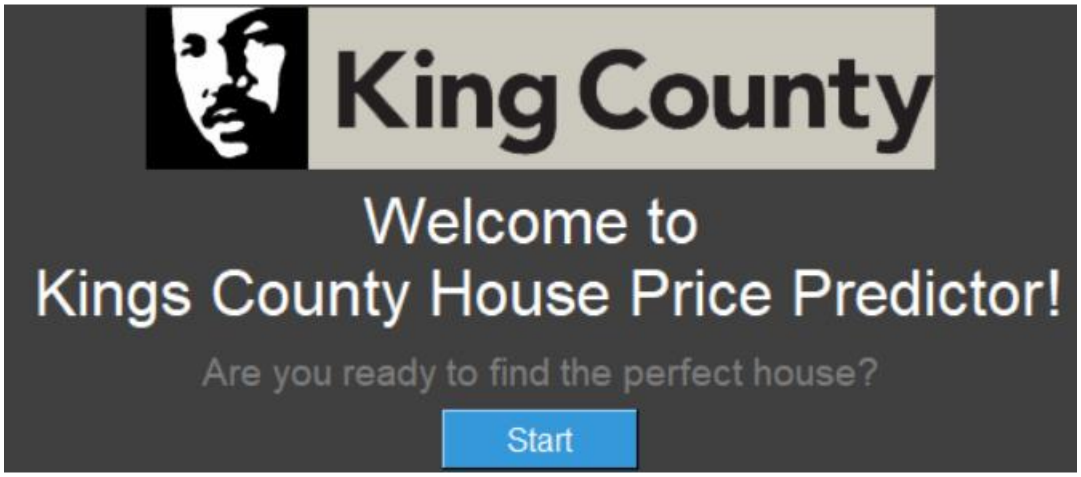
The open_window function takes in two arguments: result and sqft_liv, which represent the predicted price of the house and the square footage of the house's living area, respectively. The function calculates the price per square foot of the house by dividing result by sqft_liv, rounds the result and price_sqft, and then opens a modal window with two sg.Text elements showing the rounded result and price per square foot.

```python
def pridect_calc():
    layout = [[sg.Text('Enter the following details of your house to pridect Price:')],
        [sg.Text('No of bedrooms', size =(15, 1)), sg.Input(key='Bedrooms',size=(28, 1))],
        [sg.Text('No of bathrooms', size =(15, 1)), sg.Input(key='Bathrooms',size=(28, 1))],
        [sg.Text('Sqft living area', size =(15, 1)), sg.Input(key='living-area',size=(28, 1))],
        [sg.Text('Sqft lot area', size =(15, 1)), sg.Input(key='lot-area',size=(28, 1))],
        [sg.Text('No of floors', size =(15, 1)), sg.Input(key='floors',size=(28, 1))],
        [sg.Text('House waterfront', size =(15, 1)), sg.Checkbox("Yes",size=(11, 1),default=True)],
        [sg.Text('No of times viewed', size =(15, 1)), sg.Input(key='view',size=(28, 1))],
        [sg.Text('House Condition',size =(15, 1)),sg.Slider(range=(1, 5), default_value=1, orientation='h',size=(22.25, 15))],
        [sg.Text('House Grade',size =(15, 1)),sg.Slider(range=(3, 13), default_value=3, orientation='h',size=(22.25, 15))],
        [sg.Text('Sqft basement area', size =(15, 1)), sg.Input(key='basement-area',size=(28, 1))],
        [sg.Text('Year built', size =(15, 1)), sg.Input(key='year_built',size=(28, 1))],
        [sg.Text('Year renovated', size =(15, 1)), sg.Input(key='year_renovated',size=(28, 1))],
        [sg.Text('Lateral coordinate', size =(15, 1)), sg.Input(key='lat',size=(28, 1))],
        [sg.Text('Longitude coordinate', size =(15, 1)), sg.Input(key='long',size=(28, 1))],
        [sg.HorizontalSeparator()],
        [sg.Column(layout=[[sg.Button('Pridect',size =(10, 1),button_color=('white', '#3498db')), sg.Button('Clear',size =(10, 1)
        )]
    ]
```

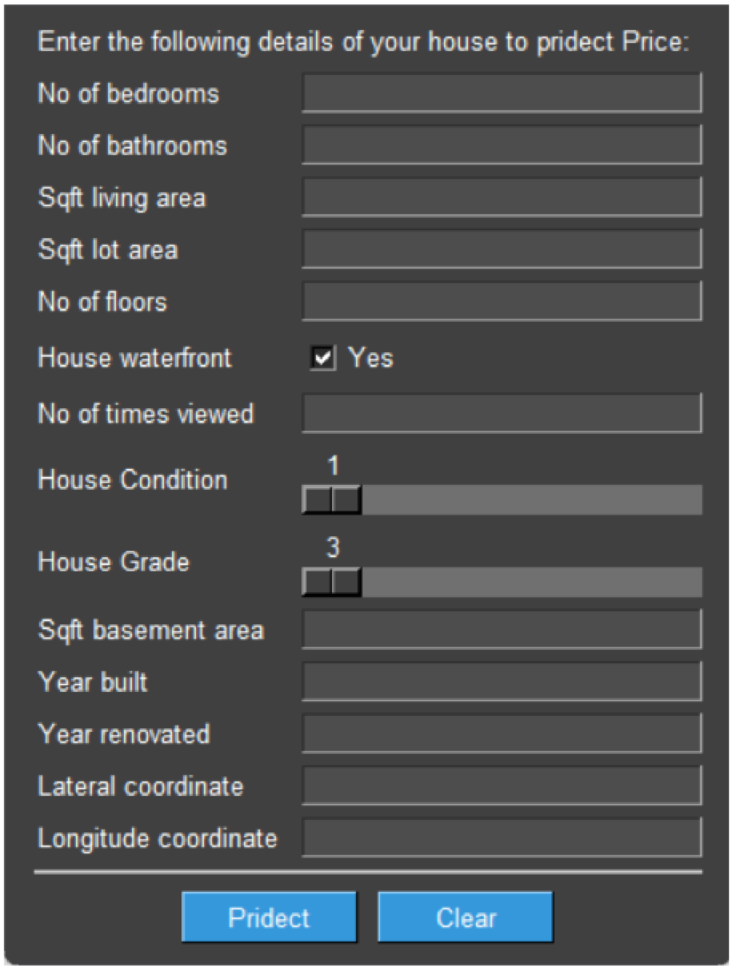
The pridect_calc function uses PySimpleGUI to create a window with a form containing a series of sg.Input elements for the user to enter various details about the house, such as the number of bedrooms, bathrooms, and square footage. The function also includes a number of sg.Checkbox, sg.Slider, and sg.Button elements for the user to interact with. The function is in an infinite loop, waiting for either the sg.WIN_CLOSED event (when the window is closed) or the

'Pridect' event (when the 'Pridect' button is clicked). If the 'Pridect' button is clicked, the function validates the input, performs some calculations using the input, and then calls the open_window function to display the result.

4- User manual:


King County
Welcome to
Kings County House Price Predictor!
Are you ready to find the perfect house?
Start

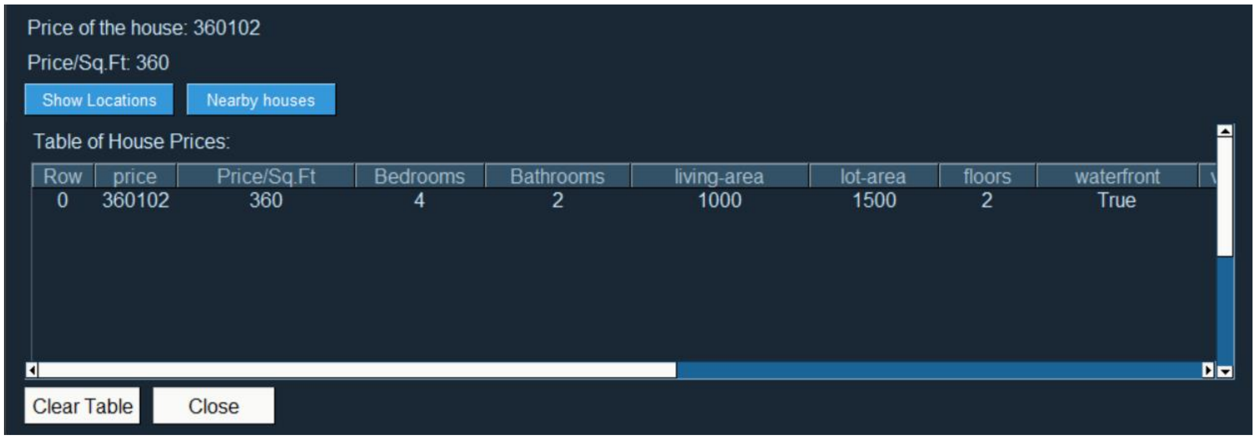
This graphical user interface is designed to help you predict the sale price of a house in Kings County based on various details about the house, such as the number of bedrooms, bathrooms, and square footage. To use this tool, simply enter the relevant information into the form on the main screen and click the 'Predict' button. The tool will then calculate the predicted sale price of the house and display it in a new window. You can also use the 'Clear' button to clear all the input fields and start over.



Enter the following details of your house to pridect Price:

No of bedrooms	
No of bathrooms	
Sqft living area	
Sqft lot area	
No of floors	
House waterfront	✓ Yes
No of times viewed	
House Condition	1
House Grade	3
Sqft basement area	
Year built	
Year renovated	
Lateral coordinate	
Longitude coordinate	

After you have entered all the relevant details about the house into the form on the main screen and clicked the 'Predict' button, a new window will appear displaying the predicted sale price of the house as well as the price per square foot. The predicted sale price of the house is calculated based on the details you have provided, and the price per square foot is calculated by dividing the predicted sale price by the square footage of the house's living area. You can close the window by clicking the 'OK' button. Thank you for using the Kings County house sale price predictor tool!


Price of the house: 360102

Price/Sq.Ft: 360

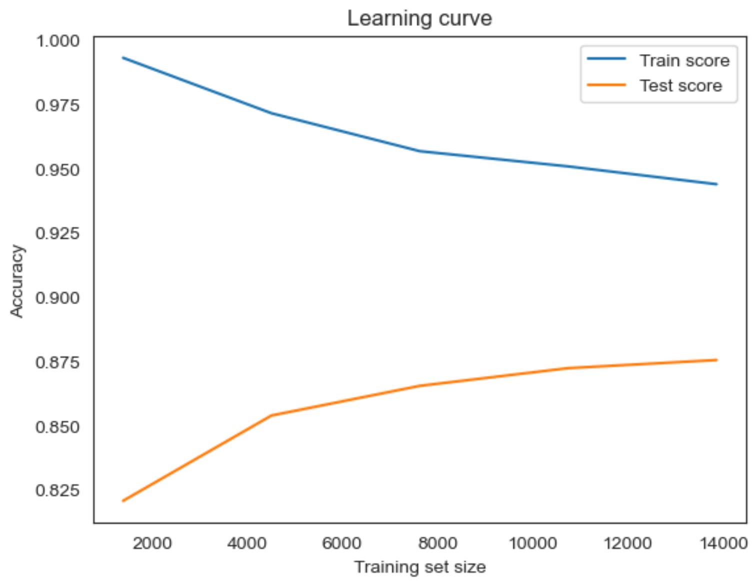
Table of House Prices:

Row	price	Price/Sq.Ft	Bedrooms	Bathrooms	living-area	lot-area	floors	waterfront
0	360102	360	4	2	1000	1500	2	True

Show locations button will show the targeted house on the map of kings county and nearby houses button will show the details of the houses near the targeted house .

5- Model results and discussion:

The model appears to be able to fit the training data well, as indicated by the high R2 score of 0.9388. However, its performance on the test data is slightly lower, with an R2 score of 0.87681. This suggests that the model may be overfitting to the training data slightly. Despite this, the model's root mean squared error (RMSE) of 49392.0239 USD is relatively low, indicating that it is able to make accurate predictions. Upon comparison with other models, our model's RMSE value is lower and its accuracy is higher.


Learning curve
Train score
Test score
Accuracy
Training set size