Build a Housing Price Predictor

StartOnAI Machine Learning Tutorial 1

This is a tutorial which navigates the process of building a Housing Price Predictor

What is a Housing Price Predictor?

A housing price predictor is meant to predict the price of a house given a feature of the house. In this tutorial, we will use linear regression to model this. This is a great project to begin with because it will help you better understand linear regression, will familiarize you with the various libraries, and provide an introduction to more complex topics.

What is Linear Regression and how does it work?

For simplicity, we will only consider one feature of the house rather than multiple and use a simple linear regression algorithm to predict a house's sale price. Regression is essentially finding the relationship between variables and linear regression is the same thing except there is one independent variable and there exists a linear relationship between the independent and dependent variable.

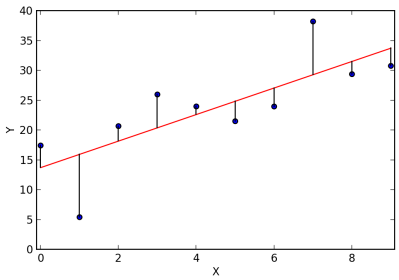

Figure 1: Linear Regression

Figure 1 contains 10 data points shown in blue and a line of best fit shown in red. The red line is what we want to predict and it is in the form $h_\theta(x) = a_0 + a_1 x$ or $y = mx + b$. The next step is how do we determine the a_0 and a_1 values so our line accurately represents the data.

To get the correct a_0 and a_1 values we use something called a cost function to determine how accurate our line is, for example, the Mean Squared Error(MSE) cost function. The cost function calculates the error of the line we will draw, so essentially we want to minimize the cost function so we have a line that is as accurate as possible. To minimize the cost function we can use an algorithm called gradient descent which we will talk about in a later lesson, but for now back to the cost function.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})^2 \tag{1}$$

The function above is the MSE function and calculates the distance from the predicted y value on the regression line and the actual y value of a data point. The gradient descent algorithm changes the values a_0 and a_1 so that the cost function reduces meaning our line accurately fits the data. See *More Resources* for a detailed look into the cost function.

How do we build a Housing Price Predictor?

1. We begin by importing the necessary libraries for this project: **numpy**, **pandas**, **seaborn** and **matplotlib**. These libraries are fairly common in machine learning and you will be using them frequently in the future. Numpy is used for working arrays. Matplotlib and Seaborn are used for data visualization. We also want to import a linear regression function from **sklearn**. Sklearn is a very useful library because it has many built-in functions that make any machine learning project much less tedious. If you aren't familiar with some of the functions used in this tutorial please look them up in the documentation, it helps if you know what the function does and the parameters that it takes.

```
In [2]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sb
        from sklearn.model_selection import train_test_split
        from sklearn.linear_model import LinearRegression
```

2. Next we download our data set as a **.csv** file and import it. For this project we will be using a data set available on *Kaggle*.

```
In [3]: data = pd.read_csv('...../kc_house_data.csv') #put your file path here
```

3. Now to understand our data a bit better we can call **.info()** and **.head()**. The **.info()** function will give us some information about the type of data

in our set. The **.head()** function gives us the first 5 rows of data, this function can pass in an integer to increase or decrease the number of rows to display. After calling the functions you should see that there are a total of 21 columns, meaning 21 features, and the first 5 rows of data. See More Resources for the output.

```
In [4]: data.info()
        data.head()
```

4. Lets select the feature we will be using next. Since we have a linear regression we must select one feature which we believe correlates well with what we want to predict, in this case, price. To determine how well a variable correlates to the price we will use the Pearson coefficient of correlation, this will give us a number between -1 and 1 which indicates how well two variables are linearly related. We create a variable called correlation and set it to **data.corr(method = 'pearson')**. This function will determine the Pearson coefficient of correlation, but right now we will see many decimals if we display this and it may be slightly difficult to understand. To make it easier on the eyes we use the functions shown below.

```
In [5]: correlation = data.corr(method = 'pearson')
        plt.figure(figsize=(21, 21))
        sb.heatmap(correlation, xticklabels = correlation.columns,
                   yticklabels = correlation.columns, annot = True,
                   linewidth = 1)
```

   Running this will display a large heatmap showing the correlations between the features we have, looking at the price column we can see that the "sqft_living" feature has a high correlation with the price so we will select this feature for our linear regression. Keep in mind that every variable will have a correlation of 1 with itself so don't be confused by the row of ones in the main diagonal. See More Resources for the output.

5. Since we don't have test data we will split our training data into training and testing data. We can do this by calling the **train_test_split** function shown below.[1] After this, we can create a variable called regress and set it the linear regression function that we imported earlier.

```
In [7]: training_data, testing_data = train_test_split(data, train_size = 0.8, random_state = 0) #splitting data
        regress = LinearRegression()
```

[1] Running this may give a FutureWarning, it is warning us that we haven't explicitly declared the test size but sklearn automatically will handle it.

6. Next we create 4 variables x_train, x_test, y_train, and y_test. The x_train variable will consist of an array, the first parameter it takes will be an object, one which exposes the array interface, and the second parameter it takes will be a datatype. We will also reshape this array by calling the **.reshape()** function which we will pass in -1 and 1 because we have a single feature. The -1 represents that we have an unknown amount of rows[2] which the function will automatically figure out for us and the 1 represents that we have one column. We do the same for the y_test variable except we don't reshape it because this is our dependent variable. We then call the **.fit()** function which takes in training data and the target value and therefore we pass in the x_test and y_test variables. We repeat the process except now we do it with the testing data.

```
In [8]: x_train = np.array(training_data['sqft_living'], dtype = pd.Series).reshape(-1,1)
        y_train = np.array(training_data['price'], dtype = pd.Series)
        regress.fit(x_train, y_train)

        x_test = np.array(testing_data['sqft_living'], dtype=pd.Series).reshape(-1,1)
        y_test = np.array(testing_data['price'], dtype=pd.Series)
```

7. Now we can predict what the line would look like on the testing data based on the training data that the **.fit()** function took in. Our final step is to graph.

```
In [ ]: prediction = regress.predict(x_test)

        plt.figure(figsize=(10,8))
        plt.plot(x_test, prediction, color = "red", label = " Regression Line")
        plt.scatter(x_test, y_test, color = 'darkgray',label="Testing Data", alpha = 0.5, edgecolors = 'black')
        plt.xlabel("sqft_living", fontsize=20)
        plt.ylabel("price", fontsize=20)
```

More Resources

1. More information on cost functions and gradient descent.
   (a) Understanding and Calculating the Cost Function for Linear Regression by Lachlan Miller
   (b) Machine Learning week 1: Cost Function, Gradient Descent and Univariate Linear Regression by Lachlan Miller
2. Similar projects with the same algorithm.
   (a) Predicting house prices with linear regression by Sara Gaspar
   (b) Regression using sklearn on KC Housing Dataset by Nikhil Kumar Mutyala

[2] We know this number from our **.info()** function we called earlier, but it would be more accurate to let python do the work and avoid any human error.

3. Code Walkthrough Ouputs

References

1. Figure 1: "Wikipedia.org." Wikipedia.org, https://commons.wikimedia.org/wiki/File:Residuals_for_Linear_Regression_Fit.png.

2. Burhan Y. Kiyakoglu. "Predicting House Prices." Kaggle, Kaggle, 23 May 2019, https://www.kaggle.com/burhanykiyakoglu/predicting-house-prices/data-Creating-a-Simple-Linear-Regression