

**Kyiv National University of Trade and Economics**

Department of Digital Economy and System Analysis

**GRADUATION QUALIFICATION WORK**

on the topic:

**“Neural network forecasting of price's dynamics of goods and services”**

Student of the 2<sup>nd</sup> year, 1m group,

specialty

051 “Economics”

specialization

“Digital Economics”



*signature of student*

Oleksandr Zorenko

Scientific supervisor

PhD in Economics,

Associate Professor

*signature of supervisor*

Volodymyr Kulazhenko

Guarantor of educational program

Doctor of Physical and

Mathematical Sciences, Professor

*signature of guarantor*

Volodymyr Hamalii

**Kyiv 2020**

**Kyiv National University of Trade and Economics**

Faculty of Information Technologies

Department of Digital Economy and System Analysis

Master's degree

Specialty 051 "Economics"

Specialization "Digital Economics"

**Approved by**

Head of the department \_\_\_\_\_ A.A. Roskladka

"15" January 2020

**Task**

**for the graduation qualification work for**

**Oleksandr Zorenko**

*(full name)*

1. Topic of the graduation qualification work:

"Neural network forecasting of price's dynamics of goods and services"

Approved by the KNUTE decree from "2" December 2019 № 4145

2. Deadline for submitting the finished work by the student "05" November 2020

3. Target setting and initial data to work:




The purpose of the study: to analyze theoretical, methodological and practical aspects of real estate price forecasting using neural network.

The object of the study: real estate price forecasting process.

The subject of the study: theoretical, methodological and practical bases of forecasting, machine learning algorithms and neural networks.

4. List of graphic material: the work contains 1 table, 31 figures, 10 formulas.

4. Advisors on work with the indication of the sections on which counseling is carried out:

Section	Advisor (full name)	Signature, date	
		Task issued	Task accepted
1	V.V. Kulazhenko	15.01.2020	 15.01.2020
2	V.V. Kulazhenko	15.01.2020	 15.01.2020
3	V.V. Kulazhenko	15.01.2020	 15.01.2020

5. Contents of the graduation qualification work (list of questions for each section)

### INTRODUCTION

#### SECTION 1. THEORETICAL FUNDAMENTALS OF REAL ESTATE PRICE FORECASTING

1.1. General characteristics of real estate and features of its pricing

1.2. The essence of traditional methods of price forecasting

Conclusions to the section 1

#### SECTION 2. ANALYSIS OF FEATURES OF NEURAL NETWORK FORECASTING OF TIME SERIES

2.1. Analysis of the machine learning workflow and data preparation

2.2. Neural networks and principles of their learning

2.3. Advantages of analyzing data and forecasting using neural networks

Conclusions to the section 2

#### SECTION 3. BUILDING NEURAL NETWORK TO FORECAST REAL ESTATE PRICES

3.1. Exploratory data analysis and feature engineering

3.2. Creating and training of the neural network model

3.3 Model usage and evaluation

Conclusions to the section 3

### CONCLUSIONS

### REFERENCES

### APPENDICES

## 6. Calendar plan of work performance

№ sequence number	Name of stages of graduation qualification work	Term work performance stages	
		according to the plan	actually
1	2	3	4
1	<i>Choosing the topic of the graduation qualification work</i>	01.12.2019	01.12.2019
2	<i>Development and approval of the task for the graduation qualification work</i>	15.01.2020	15.01.2020
3	<i>Introduction</i>	01.06.2020	
4	<i>Section 1. Theoretical fundamentals of real estate price forecasting</i>	25.06.2020	
5	<i>Section 2. Analysis of features of neural network forecasting of time series</i>	02.09.2020	
6	<i>Preparation of an article in the collection of masters' scientific articles</i>	07.09.2020	
7	<i>Section 3. Building neural network to forecast real estate prices</i>	19.10.2020	
8	<i>Conclusions</i>	02.11.2020	
9	<i>Submission of the graduation qualification work to the department to the scientific supervisor</i>	05.11.2020	
10	<i>Preliminary presentation of the graduation qualification work</i>	20.11.2020	
11	<i>Correction of remarks, external review of the graduation qualification work</i>	23.11.2020	
12	<i>Presentation of the finished stitched graduation qualification work to the department</i>	25.11.2020	
13	<i>Public presentation of the graduation qualification work</i>	According to the schedule of the EC's work	

7. Date of the task issue "15" January 2020

8. Scientific supervisor of the graduation qualification work

V.V. Kulazhenko  
(surname, initials, signature)

9. Guarantor of educational program

V.F. Hamalii  
(surname, initials, signature)

10. The task was accepted by the student



O.E. Zorenko  
(surname, initials, signature)

11. Feedback from the scientific supervisor of the graduation qualification work

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

---

Scientific supervisor of the graduation qualification work

05.11.2020

(signature, date)

Pre-presentation mark \_\_\_\_\_

(name, signature, date)

12. Conclusion on the graduation qualification work

The graduation qualification work of the student O.E. Zorenko

(surname, initials)

may be admitted to the presentation of the examination board.

Guarantor of educational program \_\_\_\_\_

V.F. Hamalii

(signature, surname, initials)

Head of the department \_\_\_\_\_

A.A. Roskladka

(signature, surname, initials)

“ \_\_\_\_\_ ”

2020

## Abstract

The graduation qualification work reveals the essence of real estate price forecasting and how to do it using machine learning and neural networks, types of forecasting methods and algorithms, neural network architecture, machine learning workflow project, data preparation and how to deal with missing values in datasets. In practical part of the work was created feedforward neural network model to predict real estate prices. As a result we received more general, robust and accurate method of price forecasting which could be used by investment companies to predict real estate price and make good investment, personal benefit to predict real estate price based on different factors to buy house at the right price.

**Keywords:** price forecasting, machine learning, neural networks, data preparation.

Випускна кваліфікаційна робота розкриває сутність прогнозування цін на нерухомість та як його здійснити за допомогою машинного навчання та нейронних мереж, типи методів прогнозування та алгоритмів, архітектуру нейронних мереж, робочий процес проекту машинного навчання, підготовку даних та методи боротьби з відсутніми значеннями у наборах даних. У практичній частині роботи була створена нейромережева модель для прогнозування цін на нерухомість. В результаті ми отримали більш загальний, надійний і точний метод прогнозування цін, який може використовуватися інвестиційними компаніями для прогнозування ціни на нерухомість та здійснення хороших інвестицій, персонального використання для прогнозування ціни на нерухомість на основі різних факторів для придбання будинку за правильною ціною

**Ключові слова:** прогнозування цін, машинне навчання, нейронні мережі, підготовка даних.

# CONTENT

INTRODUCTION .....	3
SECTION 1. Theoretical fundamentals of real estate price forecasting .....	6
1.1. General characteristics of real estate and features of its pricing .....	6
1.2. The essence of traditional methods of price forecasting.....	10
Conclusions to the section 1 .....	17
SECTION 2. Analysis of features of neural network forecasting of time series.....	18
2.1. Analysis of the machine learning workflow and data preparation.....	18
2.2. Neural networks and principles of their learning .....	25
2.3. Advantages of analyzing data and forecasting using neural networks.....	30
Conclusions to the section 2 .....	32
SECTION 3. Building neural network to forecast Real Estate prices.....	33
3.1. Exploratory data analysis and feature engineering .....	33
3.2. Creating and training the model.....	47
3.3. Model usage and evaluation .....	52
Conclusions to the section 3 .....	54
CONCLUSIONS.....	55
REFERENCES .....	57
APPENDICES .....	62

## INTRODUCTION

**The relevance of the research.** We need a proper prediction on the real estate and the houses in housing market we can see a mechanism that runs throughout the properties buying and selling buying a house will be a life time goal for most of the individual. There are lot of people making huge mistakes right now when buying properties, most of the people are buying properties unseen from the people they dont know by seeing the advertisements one of the common mistakes is buying the properties that are too expensive but its not worth it.

With the rising housing prices of the last 20 years, the appraisal of real estate has become more difficult. underlined by the large differences between listing and selling prices, the valuation process brings a level of uncertainty. Nowadays many factors affect real estate market and price housing, some even surprising, such as climate change and global pandemic. Many tradinional methods have been used in the price prediction but they make prediction based on particular set of features and difficult to expend to new factors that arising and affect real estate prices. With the advances within the field of machine learning new methods of price forecasring is open, one of which is neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. Neural network can generalize – after learning from the initial inputs and their relationships, it can infer unseen relationships on unseen data as well, thus making the model generalize and predict on unseen data.

**Analysis of recent research and publications.** In the work “Prediction of residential real estate selling prices using neural networks” by Pontus Nilsson [29] author describes methods of real estate valuation and the market, uncertainties and accuracy of traditional methods and advantages of using neural network for prediction.

Based on the work of scientist Frank Rosenblatt who developed very popular machine learning algorithm named perceptron Tushar Gupta in his study of “Deep Learning: Feedforward Neural Network” [17] describes how combining many layer of perceptrons created multilayer perceptrons or feedforward neural networks, describes



architecture of neural networks and essential theory on how they learn, such as cost function, gradient-based learning and back propagation.

**The purpose of the study** – to analyze theoretical, methodological and practical aspects of real estate price forecasting using neural networks.

The purpose of the research has caused the necessity of solving the following objectives:

- to analyze traditional methods of real estate price forecasting;
- to establish general machine learning workflow;
- to define steps for data preparation;
- to analyze what type of neural network to use and principles of their learning;
- to do feature engineering on house data to create accurate model for prediction.

**The object of the study** – real estate price forecasting process.

**The subject of the study** – theoretical, methodological and practical bases of forecasting, machine learning algorithms and neural networks.

**The empirical basis of the research** – research papers, books and digital resources on real estate price prediction and artificial neural network models.

**Research methods.** The dialectical method is used in the work – the method of scientific abstraction is applied, which implies the selection of typical, stable tendencies in the researched process or phenomenon; the method of induction – theoretical conclusions, generalizations based on the study of individual facts, calculations are formulated. The method of formalization is used – the study of objects by displaying their structure in sign form using the language of mathematics. Methods of analysis and synthesis are used – the separation of the constituent parts of the subject, their study and further combination into a whole, taking into account the relationships between them. The calculation-analytical method is applied – formulas were used, and mathematical calculations were performed.

**Information support** of the research consists of scientific articles, monographs, magazine publications, textbooks, statistical data, information websites, other scientific

publications. Neural network model was created using jupyter notebook, tensorflow 2.0 and Keras api.

**The practical significance of the study.** The results of the study are of practical importance and could be used by investment companies to predict real estate price and make good investment and personal benefit to predict real estate price based on different factors to buy house at the right price.

**Approbation of research results.** According to the results of the research, a report was presented at the conference of KNTEU students “Scientific research of student youth”, section “Applied aspects of information technology” on the topic: “Forecasting of price dynamic for goods and services using neural networks”, which took place on April 7-8, 2020.

According to the results of the conducted research, a scientific article on the topic “Forecasting of price dynamic for goods and services using neural networks” was published, which was included in the collection of scientific articles of KNUTE students “Digital Economy”, Kyiv 2020.

**Work structure.** The work consists of an introduction, three sections, conclusions, a list of references and appendices, which contain (example of data from dataset and full progress of development neural network model). The total volume of the work is 81 pages. It contains 1 table, 31 figures, 10 formulas. Number of used references – 38, their list is given on 5 pages.

## SECTION 1

### Theoretical fundamentals of real estate price forecasting

#### 1.1. General characteristics of real estate and features of its pricing

Real estate is the land along with any permanent improvements attached to the land, whether natural or man-made—including water, trees, minerals, buildings, homes, fences, and bridges. Real estate is a form of real property. It differs from personal property, which are things not permanently attached to the land, such as vehicles, boats, jewelry, furniture, and farm equipment [1].

There are four types of real estate:

1. Residential real estate includes both new construction and resale homes. The most common category is single-family homes. There are also condominiums, co-ops, townhouses, duplexes, triple-deckers, quadplexes, high-value homes, multi-generational and vacation homes.
2. Commercial real estate includes shopping centers and strip malls, medical and educational buildings, hotels and offices. Apartment buildings are often considered commercial, even though they are used for residences. That's because they are owned to produce income.
3. Industrial real estate includes manufacturing buildings and property, as well as warehouses. The buildings can be used for research, production, storage, and distribution of goods. Some buildings that distribute goods are considered commercial real estate. The classification is important because the zoning, construction, and sales are handled differently.
4. Land includes vacant land, working farms, and ranches. The subcategories within vacant land include undeveloped, early development or reuse, subdivision and site assembly [3].

Real estate also has some distinct economic characteristics that influence its value as an investment. There are four economic characteristics:

### 1. *Scarcity*

While land isn't considered rare, the total supply is fixed. This can have a significant impact on the value of a property. Especially in highly populous areas, such as Long Island. The scarcer the land, the higher the price. It is a supply and demand concept [3].

### 2. *Improvements*

The economic characteristic of improvements (also known as modification), states that improvements to a piece of land can have either a positive or negative impact on its value. Adding a pool and landscaping to a home will increase its value. If a nuclear power plant is built, the surrounding land values will decline [3].

Improvements of a private nature (such as homes and fences) are referred to as improvements on the land. Improvements of a public nature (e.g., sidewalks and sewer systems) are called improvements to the land [2].

### 3. *Permanence of investment*

Permanence of investment is also known as fixity and means investments in real estate are long-term. This is due to the physical characteristics of indestructibility and immobility. Since land is immobile, investment in property becomes fixed. Land cannot be moved if the market becomes better in another location [3].

Once land is improved, the total capital and labor used to build the improvement represent a sizable fixed investment. Even though a building can be razed, improvements like drainage, electricity, water, and sewer systems tend to be permanent because they can't be removed (or replaced) economically [2].

Since real estate transactions are complex and large amounts of money are involved, they are not made very frequently. As a result, a real estate investment is a long-term investment.

### 4. *Area preference*

Area preference is the most important economic characteristics of land. Situs is based on many factors, such as history, convenience, and reputation. A home in a

neighborhood with great schools and a low crime rate will generally command a higher price. A house in a high crime neighborhood with poor schooling would be priced lower. Location is one of the most important economic characteristics of land [2].

Despite the magnitude and complexity of the real estate market, many people tend to think the industry consists merely of brokers and salespeople. However, millions of people in fact earn a living through real estate, not only in sales but also in appraisals, property management, financing, construction, development, counseling, education, and several other fields [1].

Many professionals and businesses – including accountants, architects, banks, title insurance companies, surveyors, and lawyers – also depend on the real estate industry[1].

Real estate is a critical driver of economic growth in the U.S. In fact, housing starts – the number of new residential construction projects in any given month – released by the U.S. Census Bureau is a key economic indicator. Building permits, housing starts, and housing completions data, divided into three different categories:

- single-family homes;
- homes with 2-4 units;
- multifamily buildings with five or more units, such as apartment complexes.

Investors and analysts keep a close eye on housing starts because the numbers can provide a general sense of economic direction. Moreover, the *types* of new housing start can give clues about how the economy is developing [1, 3].

Example: Housing Starts.

For example, if housing starts indicating fewer single-family and more multifamily starts, it could indicate an impending supply shortage for single-family homes – which could drive up home prices. The following chart shows 20 years of housing starts, from Jan. 1, 2000, to Feb. 1, 2020 [1].



Figure 1.1. 20 years of housing starts

Source: designed by the author based on [1]

*Value Versus Cost and Price.* Value is not necessarily equal to cost or price. Cost refers to actual expenditures – on materials, for example, or labor. Price, on the other hand, is the amount that someone pays for something. While cost and price can affect value, they do not determine value. The sales price of a house might be \$150,000, but the value could be significantly higher or lower. For instance, if a new owner finds a serious flaw in the house, such as a faulty foundation, the value of the house could be lower than the price [5].

*Market Value.* An appraisal is an opinion or estimate regarding the value of a particular property as of a specific date. Appraisal reports are used by businesses, government agencies, individuals, investors, and mortgage companies when making decisions regarding real estate transactions. The goal of an appraisal is to determine a property's market value – the most probable price that the property will bring in a competitive and open market.

Market price, the price at which property actually sells, may not always represent the market value. For example, if a seller is under duress because of the threat of foreclosure, or if a private sale is held, the property may sell below its market value [5].

## **1.2. The essence of traditional methods of price forecasting**

Estimating the value of real estate is necessary for a variety of endeavors, including financing, sales listing, investment analysis, property insurance, and taxation. But for most people, determining the asking or purchase price of a piece of real property is the most useful application of real estate valuation [5].

Property valuation is important to calculate and understand before the purchasing of a property. While some signs, like location and square footage of a property, are important to finding out the ultimate value of the property, they can also be misleading. These nuances can make a potential property seem like a better investment than it actually is [4].

Using calculations and careful estimates based on the values of the neighboring properties can help to see if a property for potential investment will meet investment goals.

There are 3 traditional methods to run a valuation on property:

- comparable sales approach;
- income approach;
- cost approach [4].

### *Method 1: Sales Comparison Approach*

The sales comparison approach is commonly used in valuing single-family homes and land. Sometimes called the market data approach, it is an estimate of value derived by comparing a property with recently sold properties with similar characteristics. These similar properties are referred to as comparable, and in order to provide a valid comparison, each must:

- be as similar to the subject property as possible;
- have been sold within the last year in an open, competitive market;

- have been sold under typical market conditions.

At least three or four comparables should be used in the appraisal process. The most important factors to consider when selecting comparables are the size, comparable features and – perhaps most of all – location, which can have a tremendous effect on a property's market value [5].

#### *Comparables` Qualities*

Since no two properties are exactly alike, adjustments to the comparables' sales prices will be made to account for dissimilar features and other factors that would affect value, including:

- age and condition of buildings;
- date of sale, if economic changes occur between the date of sale of a comparable and the date of the appraisal;
- terms and conditions of sale, such as if a property's seller was under duress or if a property was sold between relatives (at a discounted price);
- location, since similar properties might differ in price from neighborhood to neighborhood;
- physical features, including lot size, landscaping, type and quality of construction, number and type of rooms, square feet of living space, hardwood floors, a garage, kitchen upgrades, a fireplace, a pool, central air, etc.

The market value estimate of the subject property will fall within the range formed by the adjusted sales prices of the comparables. Since some of the adjustments made to the sales prices of the comparables will be more subjective than others, weighted consideration is typically given to those comparables that have the least amount of adjustment [5].

Comparable sales approach involves the following steps:

1. Identifying actual market transactions in recent past of comparable properties. The comparable transactions must be similar in terms of location, property size, property nature (residential vs commercial), tenant size (whether one tenant or multiple tenants), typical lease duration (i.e. short-term vs long-term, etc.).



2. Finding price multiple for the properties based on some feature of the property which derives the property's value. It involves dividing the property value by say its covered area, number of apartments, etc.
3. The multiples derived from comparable transactions are then multiplied with the same feature of the property to arrive at a value estimate, i.e. value per square feet must be multiplied by the property's area in square feet to find value [6].

For example, let's assume we own a 50-unit residential 3-bed apartment complex in Toronto with total area of 50,000 square feet. We are going to call it Property Y.

Average occupancy rate is 90% and the average monthly rent is CAD 8,000 expected to grow by 7% each year for foreseeable future i.e. initial 5 years. 6% of the revenue is never collected. Monthly operating costs are 20% of the revenue in first year and are expected to grow in line with the Consumer Price Index (CPI), let's say 3% per annum. Insurance costs are 3% of the capacity. Other revenue per year amounts to \$500,000 which are expected to stay constant for next 5-years. Property taxes are 5% and income tax is 20%. During last year, three apartment buildings were sold within one square kilometer area: Building A had a total covered area of 25,000 square feet building, had 30 units and was sold for \$40 million; Building B had area of 70,000 square feet, 60 units and was valued at \$70 million and Building C had area of 60,000 square feet, 42 apartment and was sold at \$55 million [6, 7].

The market value of the land is \$30 million, and it took \$20 million to construct the building 10 years ago. Assuming inflation over the last 10 years to be 2.5% on average. During the last 3 years, comparable properties were valued based on a 10% capitalization rate. Assume in a net average growth in NOI is 5% for the purpose of direct capitalization method forever. Work out a lower and upper bound for the property's value based on the most popular real estate valuation methods. Required rate of return keeping in view the risk is 11% [6, 7].

The comparable sales approach is the simplest method even though it is impossible to find a perfectly comparable property. The following table shows how the comparable sales approach can be applied to the Property Y [6].

Table 1.1

*Factors of the enterprise's intellectual potential development*

Property	Value	Area (Square ft)	Units	Value per Square ft	Value per unit
A	40,000,000	25000	30	1,600	1,333,333
B	70,000,000	70000	60	1,000	1,166,667
C	55,000,000	60000	42	917	1,309,524
<b>Average</b>				1,172	1,269,841
<b>Property Y area (Square ft)</b>				50,000	Property Y area (Square ft)
<b>Property Y units</b>					50
<b>Property Y value</b>				\$58,611,111	\$63,492,063

Source: designed by the author based on [6]

*Method 2: Income approach*

The income capitalization approach, or income approach, is a valuation of real estate commonly used for rental properties and commercial real estate properties. This method converts the income of a property into an estimate of its value.

The general formula for calculating the valuation of a property, also known as IRV in this method, is as follows:

$$\text{Net operating income (I) / capitalization rate (R) = value (V),} \quad (1.1)$$

Breaking this calculation into a few extra steps is helpful. First calculate the NOI (Net Operating Income), as follows:

*1. Calculate the annual potential gross income*

The potential gross income is the potential rental income of the property when rented at 100% capacity.

For example, if the monthly rent is \$1,000 then your annual potential gross income is  $12 \times \$1,000 = \$12,000$  [4].

*2. Calculate the effective gross income*

This number, which usually is expressed as a percentage, is the appraiser's estimate from the market for these kinds of buildings in the local area. The effective gross income is the potential gross rental income plus other income minus the vacancy rate and credit costs[4].

For example, the vacancy rate of property could be 5% and the additional income might be \$100 per month, or \$1,200 annually.

At this point: A property with a potential gross income of \$12,000 - 5% vacancy (or \$600) + additional income (or \$1,200) = \$12,600 [7].

### *3. Calculate the net operating income (NOI)*

Start by deducting annual operating expenses such as real estate and personal property taxes, property insurance, management fees (on or off-site), repairs and maintenance, utilities, and other miscellaneous expenses (accounting, legal, etc.) [4].

$$\text{Effective gross income} - \text{operating expenses} = \text{NOI}, \quad (1.2)$$

At this point: Our Effective gross income is \$12,600 for this property. Let's say all the additional operating expenses are \$10,100 for the property. This means the NOI is \$2,500.

Now that you have your NOI calculated, you can continue on to estimate the valuation of your chosen property [8. p. 4].

### *4. Compare similar cap rates*

A capitalization rate is similar to a rate of return; that is, the percentage that the investors hope to get out of the building in income.

Look at similar properties' cap rates to estimate the price an investor would pay for the income generated by the particular property.

Lofty AI's ROI on rental property calculator allows you to calculate cash on cash return, cash flow, and is also a cap rate calculator.

Let's choose a cap rate of 10% for this example [4].

### *5. Apply the cap rate to the property's annual NOI*

This last step allows you to form an estimate of the property's value, and where the formula is used.

All you have to do now is divide the NOI by the cap rate.

To finish the example:  $\$2,500 / 0.10 = \$25,000$

\$25,000 is the estimate of the valuation of this property, using the income capitalization approach[4, 7].

### *Method 3: Cost Approach*

The cost approach can be used to estimate the value of properties that have been improved by one or more buildings. This method involves separate estimates of value for the building(s) and the land, taking into consideration depreciation. The estimates are added together to calculate the value of the entire improved property. The cost approach makes the assumption that a reasonable buyer would not pay more for an existing improved property than the price to buy a comparable lot and construct a comparable building. This approach is useful when the property being appraised is a type that is not frequently sold and does not generate income. Examples include schools, churches, hospitals and government buildings [5].

Building costs can be estimated in several ways, including the square-foot method where the cost per square foot of a recently built comparable is multiplied by the number of square feet in the subject building; the unit-in-place method, where costs are estimated based on the construction cost per unit of measure of the individual building components, including labor and materials; and the quantity-survey method, which estimates the quantities of raw materials that will be needed to replace the subject building, along with the current price of the materials and associated installation costs [9].

### *Depreciation*

For appraisal purposes, depreciation refers to any condition that negatively affects the value of an improvement to real property, and takes into consideration:

- physical deterioration, including curable deterioration, such as painting and roof replacement, and incurable deterioration, such as structural problems;
- functional obsolescence, which refers to physical or design features that are no longer considered desirable by property owners, such as outdated appliances, dated-looking fixtures or homes with four bedrooms, but only one bath;
- economic obsolescence, caused by factors that are external to the property, such as being located close to a noisy airport or polluting factory [5, 6].

*Methodology:*

- estimate the value of the land as if it were vacant and available to be put to its highest and best use, using the sales comparison approach since land cannot be depreciated;
- estimate the current cost of constructing the building(s) and site improvements;
- estimate the amount of depreciation of the improvements resulting from deterioration, functional obsolescence or economic obsolescence;
- deduct the depreciation from the estimated construction costs;
- add the estimated value of the land to the depreciated cost of the building(s) and site improvements to determine the total property value [5, 6, 7].

When using this method, it is important to remember that it can be helpful when calculating the valuation of a property, but it does not take into account the surrounding factors or the factors of that specific property, which as we have previously said, can skew your results dramatically.

The most commonly used cost approach appraisals include:

- *Reproduction cost* - The cost to construct an exact duplicate of the subject property at today's costs;
- *Replacement cost* - The cost to construct a structure with the same usefulness (utility) as a comparable structure using today's materials and standards [4].

When all estimates have been gathered, the cost approach is calculated in the following way:

- *Replacement or Reproduction Cost* – Depreciation + Land Worth = Value of the Property [4].

Although these methods are easy to use and took their place as traditional methods of real estate valuation, they are difficult to apply in densely populated urban areas where sales or rentals of unimproved land are rare. Existence of depreciation, or any deviation from highest and best use that would distort the income available to the unimproved land, can leave the independent value of the improvements extremely uncertain. Physical, economic or functional depreciation greatly complicates the

attempt to calculate building value. For more general and complex price prediction better solution would be use of neural networks. After learning from the initial inputs and their relationships, neural network can infer unseen relationships on unseen data, thus making the model generalize and predict on unseen data.

### **Conclusions to the section 1**

In the following section theoretical and methodological aspects of real estate were analyzed. After analyzing scientific literature, we have defined economic characteristics of real estate that influence its value as an investment.

Estimating the value of real estate is necessary for a variety of endeavors, including financing, sales listing, investment analysis, property insurance, and taxation. But for most people, determining the asking or purchase price of a piece of real property is the most useful application of real estate valuation. We have determined 3 traditional methods to run a valuation on property: comparable sales approach, Income approach, cost approach

Accurate real estate valuation is important to mortgage lenders, investors, insurers and buyers, and sellers of real property. While appraisals are generally performed by skilled professionals, anyone involved in a real transaction can benefit from gaining a basic understanding of the different methods of real estate valuation.

## SECTION 2

### Analysis of features of neural network forecasting of time series

#### 2.1. Analysis of the machine learning workflow and data preparation

*Price forecasting* is predicting a commodity/product/service price by evaluating various factors like its characteristics, demand, seasonal trends, other commodities' prices (i.e. fuel), offers from numerous suppliers, etc.

*Price prediction* can be formulated as a regression task. Regression analysis is a statistical technique used to estimate the relationship between a dependent/target variable (electricity price, flight fare, property price, etc.) and single or multiple independent (interdependent) variables AKA predictors that impact the target variable. Regression analysis also lets researchers determine how much these predictors influence a target variable. In regression, a target variable is always numeric [10].

In general, price forecasting is done by the means of descriptive and predictive analytics. Descriptive analytics.

*Descriptive analytics* rely on statistical methods that include data collection, analysis, interpretation, and presentation of findings. Descriptive analytics allow for transforming raw observations into knowledge one can understand and share. In short, this analytics type helps to answer the question of what happened [10, 11].

*Predictive analytics*. Predictive analytics is about analyzing current and historical data to forecast the probability of future events, outcomes, or values in the context of price predictions. Predictive analytics requires numerous statistical techniques, such as data mining (identification of patterns in data) and machine learning [10, 11].

The goal of machine learning is to build systems capable of finding patterns in data, learning from it without human intervention and explicit reprogramming. To solve the price prediction problem, data scientists first must understand what data to use to train machine learning models, and that's exactly why descriptive analytics is needed [12].

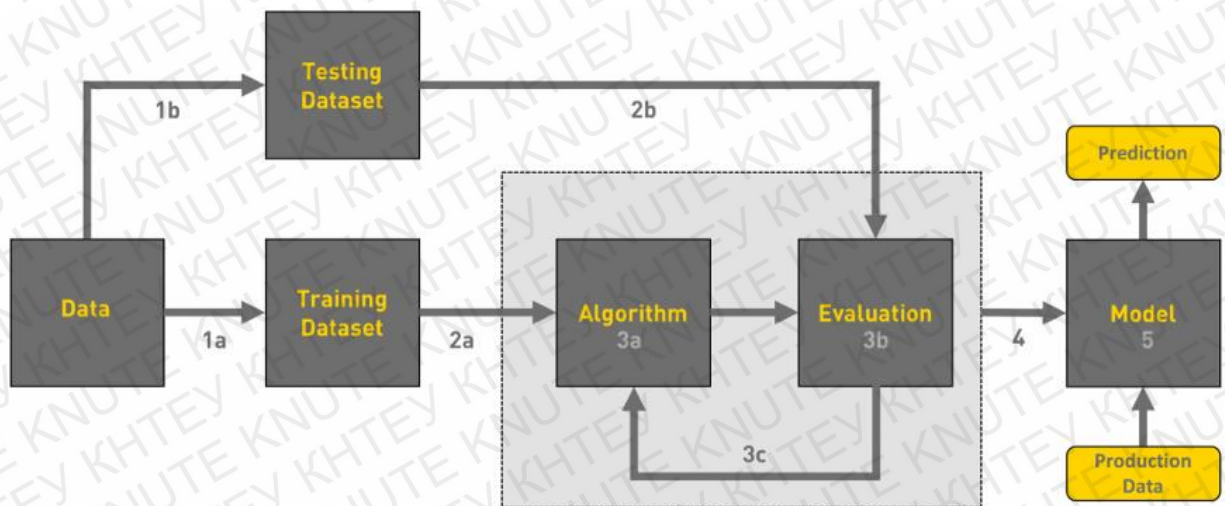


Figure 2.1. Machine learning workflow

Source: designed by the author based on [13]

While ML projects vary in scale and complexity requiring different data science teams, their general structure is the same. For example, a small data science team would have to collect, preprocess, and transform data, as well as train, validate, and (possibly) deploy a model to do a single prediction [14].

I would like to briefly describe a common scenario for ML project implementation:

1. *Strategy*: matching the problem with the solution. In the first phase of an ML project realization, company representatives mostly outline strategic goals. They assume a solution to a problem, define a scope of work, and plan the development. For example, your eCommerce store sales are lower than expected. The lack of customer behavior analysis may be one of the reasons you are lagging behind your competitors.

2. *Dataset preparation and preprocessing*. Data is the foundation for any machine learning project. The second stage of project implementation is complex and involves data collection, selection, preprocessing, and transformation [15].

- *Data collection*

There is no exact answer to the question “How much data is needed?” because each machine learning problem is unique. In turn, the number of attributes data scientists will use when building a predictive model depends on the attributes’ predictive value.

‘The more, the better’ approach is reasonable for this phase. Some data scientists suggest considering that less than one-third of collected data may be useful. It’s difficult



to estimate which part of the data will provide the most accurate results until the model training begins. That's why it's important to collect and store all data — internal and open, structured and unstructured [12].

The tools for collecting internal data depend on the industry and business infrastructure. For example, those who run an online-only business and want to launch a personalization campaign can try out such web analytic tools as Mixpanel, Hotjar, CrazyEgg, well-known Google analytics, etc. A web log file, in addition, can be a good source of internal data. It stores data about users and their online behavior: time and length of visit, viewed pages or objects, and location [13].

Companies can also complement their own data with publicly available datasets. For instance, Kaggle, Github contributors, AWS provide free datasets for analysis.

- *Data selection*

After having collected all information, a data analyst chooses a subgroup of data to solve the defined problem. For instance, if you save your customers' geographical location, you don't need to add their cell phones and bank card numbers to a dataset. But purchase history would be necessary. The selected data includes attributes that need to be considered when building a predictive model [14].

- *Data preprocessing*

The purpose of preprocessing is to convert raw data into a form that fits machine learning. Structured and clean data allows a data scientist to get more precise results from an applied machine learning model. The technique includes data formatting, cleaning, and sampling.

*Data formatting.* The importance of data formatting grows when data is acquired from various sources by different people. The first task for a data scientist is to standardize record formats. A specialist checks whether variables representing each attribute are recorded in the same way. Titles of products and services, prices, date formats, and addresses are examples of variables. The principle of data consistency also applies to attributes represented by numeric ranges [12].

*Data cleaning.* This set of procedures allows for removing noise and fixing inconsistencies in data. A data scientist can fill in missing data using imputation

techniques, e.g. substituting missing values with mean attributes. A specialist also detects outliers — observations that deviate significantly from the rest of distribution. If an outlier indicates erroneous data, a data scientist deletes or corrects them if possible. This stage also includes removing incomplete and useless data objects [16].

3. *Dataset splitting*. A dataset used for machine learning should be partitioned into three subsets - training, test, and validation sets:

- *Training set*. A data scientist uses a training set to train a model and define its optimal parameters – parameters it has to learn from data.
- *Test set*. A test set is needed for an evaluation of the trained model and its capability for generalization. The latter means a model’s ability to identify patterns in new unseen data after having been trained over a training data. It’s crucial to use different subsets for training and testing to avoid model overfitting, which is the incapacity for generalization we mentioned above [13].
- *Validation set*. The purpose of a validation set is to tweak a model’s hyperparameters – higher-level structural settings that can’t be directly learned from data. These settings can express, for instance, how complex a model is and how fast it finds patterns in data. The proportion of a training and a test set is usually 80 to 20 percent respectively. A training set is then split again, and its 20 percent will be used to form a validation set. At the same time, machine learning practitioner Jason Brownlee suggests using 66 percent of data for training and 33 percent for testing. A size of each subset depends on the total dataset size [14].

4. *Modeling*. After a data scientist has preprocessed the collected data and split it into three subsets, he can proceed with a model training. This process entails “feeding” the algorithm with training data. An algorithm will process data and output a model that is able to find a target value (attribute) in new data – an answer you want to get with predictive analysis. The purpose of model training is to develop a model [12].

5. *Model evaluation and testing*. The goal of this step is to develop the simplest model able to formulate a target value fast and well enough. A data scientist can achieve this goal through model tuning. That’s the optimization of model parameters to achieve

an algorithm's best performance. One of the more efficient methods for model evaluation and tuning is cross-validation. It entails splitting a training dataset into ten equal parts (folds). A given model is trained on only nine folds and then tested on the tenth one (the one previously left out). Training continues until every fold is left aside and used for testing [14].

As a result of model performance measure, a specialist calculates a cross-validated score for each set of hyperparameters. A data scientist trains models with different sets of hyperparameters to define which model has the highest prediction accuracy. The cross-validated score indicates average model performance across ten hold-out folds [15].

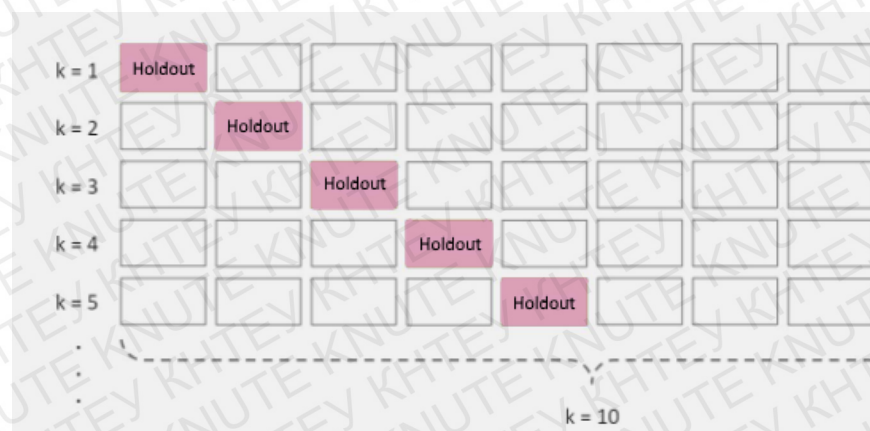


Figure 2.2. Cross-validation method of model evaluation

Source: designed by the author based on [12]

6. *Model deployment.* The model deployment stage covers putting a model into production use.

Many real-world datasets may contain missing values for various reasons. They are often encoded as NaNs, blanks or any other placeholders. Training a model with a dataset that has a lot of missing values can drastically impact the machine learning model's quality. Some algorithms such as scikit-learn estimators assume that all values are numerical and have and hold meaningful value [12]. So, we will focus on popular ways for data imputation:

- *Imputation Using (Mean/Median) Values:* This works by calculating the mean/median of the non-missing values in a column and then replacing the missing

values within each column separately and independently from the others. It can only be used with numeric data [11].

	col1	col2	col3	col4	col5		col1	col2	col3	col4	col5	
0	2	5.0	3.0	6	NaN	mean()	0	2.0	5.0	3.0	6.0	7.0
1	9	NaN	9.0	0	7.0		1	9.0	11.0	9.0	0.0	7.0
2	19	17.0	NaN	9	NaN		2	19.0	17.0	6.0	9.0	7.0

Figure 2.3. Mean Imputation

Source: designed by the author based on [11]

Pros:

- easy and fast;
- works well with small numerical datasets.

Cons:

- doesn't factor the correlations between features. It only works on the column level;
  - will give poor results on encoded categorical features (do NOT use it on categorical features);
  - not very accurate;
  - doesn't account for the uncertainty in the imputations.
- *Imputation Using (Most Frequent) or (Zero/Constant) Values:* Most Frequent is another statistical strategy to impute missing values and It works with categorical features (strings or numerical representations) by replacing missing data with the most frequent values within each column [11].

Pros:

- works well with categorical features.

Cons:

- it also doesn't factor the correlations between features;
- it can introduce bias in the data.

- *Imputation Using k-NN*: The k nearest neighbors is an algorithm that is used for simple classification. The algorithm uses ‘feature similarity’ to predict the values of any new data points. This means that the new point is assigned a value based on how closely it resembles the points in the training set. This can be very useful in making predictions about the missing values by finding the k’s closest neighbors to the observation with missing data and then imputing them based on the non-missing values in the neighborhood. It creates a basic mean impute then uses the resulting complete list to construct a KDTree. Then, it uses the resulting KDTree to compute nearest neighbors (NN). After it finds the k-NNs, it takes the weighted average of them [11].

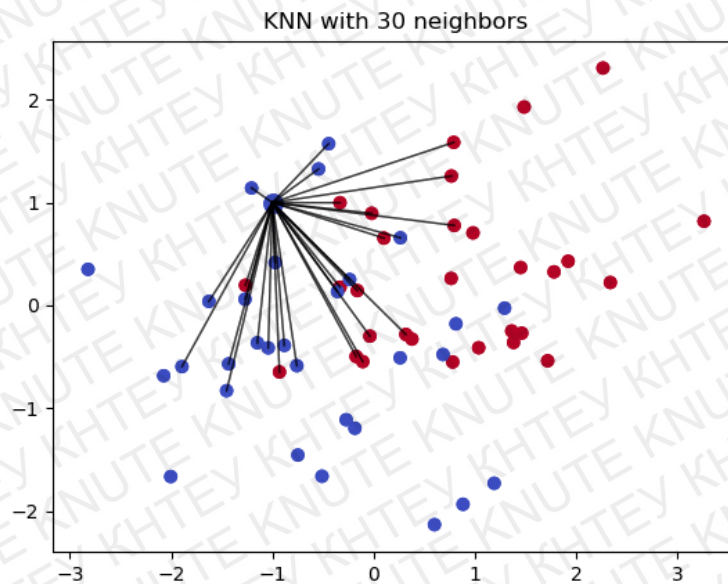


Figure 2.4. Imputation using k-NN

Source: designed by the author based on [11]

Pros:

- can be much more accurate than the mean, median or most frequent imputation methods (It depends on the dataset).

Cons:

- computationally expensive. KNN works by storing the whole training dataset in memory;
- K-NN is quite sensitive to outliers in the data [11].

## 2.2. Neural networks and principles of their learning

We will be looking at *Feedforward neural networks* (multilayer perceptron's). A neural network is a series of algorithms that endeavors to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates. In this sense, neural networks refer to systems of neurons, either organic or artificial in nature. Neural networks can adapt to changing input; so, the network generates the best possible result without needing to redesign the output criteria.

Deep feedforward networks also often called feedforward neural networks, or multilayer perceptron's (MLPs), are the quintessential deep learning models. The goal of a feedforward network is to approximate some function  $f^*$ . For example, for a classifier,  $y = f^*(x)$  maps an input  $x$  to a category  $y$ . A feedforward network defines a mapping  $y = f(x; \theta)$  and learns the value of the parameters  $\theta$  that result in the best function approximation. These models are called feedforward because information flows through the function being evaluated from  $x$ , through the intermediate computations used to define  $f$ , and finally to the output  $y$ . There are no feedback connections in which outputs of the model are fed back into itself [17].

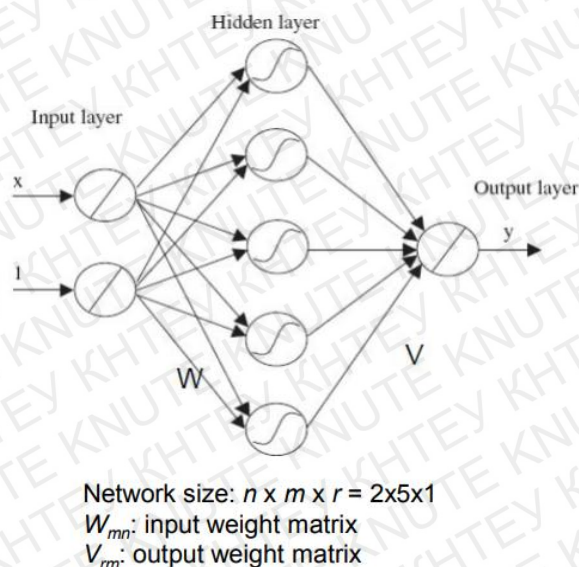


Figure 2.5. Feedforward networks

Source: designed by the author based on [19, p. 9]

*Input layer:* Number of neurons in this layer corresponds to the number of inputs to the neuronal network. This layer consists of passive nodes, i.e., which do not take part in the actual signal modification, but only transmits the signal to the following layer [19].

*Hidden layer:* This layer has arbitrary number of layers with arbitrary number of neurons. The nodes in this layer take part in the signal modification, hence, they are active.

*Output layer:* The number of neurons in the output layer corresponds to the number of the output values of the neural network. The nodes in this layer are active ones.

If a multilayer perceptron has a linear activation function in all neurons, that is, a linear function that maps the weighted inputs to the output of each neuron, then linear algebra shows that any number of layers can be reduced to a two-layer input-output model. In MLPs some neurons use a nonlinear activation function that was developed to model the frequency of action potentials, or firing, of biological neurons [18].

The two historically common activation functions are both sigmoid, and are described by

$$y(v_i) = \tanh(v_i) \text{ and } y(v_i) = (1 + e^{-v_i})^{-1}, \quad (2.1)$$

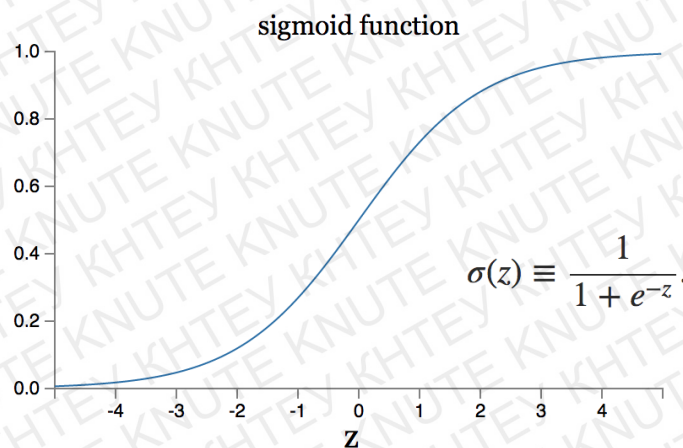


Figure 2.6. Sigmoid Function

Source: designed by the author based on [18]

Let first see the behavior of sigmoid function Figure 2.6. Unlike piecewise linear units, sigmoidal units saturate across most of their domain – they saturate to a high value

when  $z$  is very positive, saturate to a low value when  $z$  is very negative, and are only strongly sensitive to their input when  $z$  is near 0.

*Learning occurs* in the perceptron by changing connection weights after each piece of data is processed, based on the amount of error in the output compared to the expected result. This is an example of supervised learning, and is carried out through backpropagation, a generalization of the least mean squares algorithm in the linear perceptron. We can represent the degree of error in an output node  $j$  in the  $n$ -th data point (training example) by

$$e_j(n) = d_j(n) - y_j(n), \quad (2.2)$$

where  $d$  is the target value and  $y$  is the value produced by the perceptron [17]. The node weights can then be adjusted based on corrections that minimize the error in the entire output, given by

$$\varepsilon(n) = \frac{1}{2} \sum_j e_j^2(n), \quad (2.3)$$

*Gradient-Based Learning* - designing and training a neural network is not much different from training any other machine learning model with gradient descent [17]. The largest difference between the linear models we have seen so far and neural networks is that the nonlinearity of a neural network causes most interesting loss functions to become non-convex. This means that neural networks are usually trained by using iterative, gradient-based optimizers that merely drive the cost function to a very low value, rather than the linear equation solvers used to train linear regression models or the convex optimization algorithms with global convergence guarantees used to train logistic regression or SVMs [17].

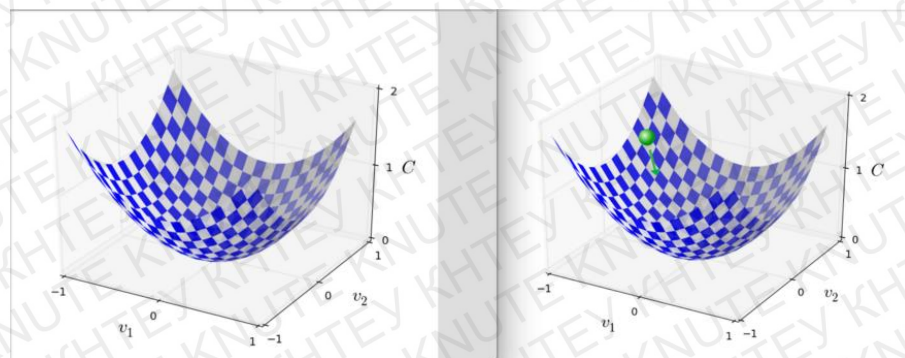


Figure 2.7. Visualization of how gradient based learning occur

Source: designed by the author based on [17]



As we can see at right of Figure 2.7., an analogy of a ball dropped in a deep valley and it settle downs at the bottom of the valley. When we move the ball a small amount  $\Delta v_1$  in the  $v_1$  direction, and a small amount  $\Delta v_2$  in the  $v_2$  direction. Calculus tells us that  $C$  changes as follows: [17, 18]

$$\Delta C \approx \frac{\partial C}{\partial v_1} \Delta v_1 + \frac{\partial C}{\partial v_2} \Delta v_2, \quad (2.4)$$

Change of the  $v_1$  and  $v_2$  such that the change in cost is negative is desirable. We can also denote  $\Delta C \approx \nabla C \cdot \Delta v$ . where  $\nabla C$  is,

$$\nabla C = \left( \frac{\partial C}{\partial v_1}, \dots, \frac{\partial C}{\partial v_m} \right)^T, \quad (2.5)$$

and  $\Delta v$  is,

$$\Delta v = (\Delta v_1, \dots, \Delta v_m)^T, \quad (2.6)$$

Indeed, there's even a sense in which gradient descent is the optimal strategy for searching for a minimum. Let's suppose that we're trying to make a move  $\Delta v$  in position so as to decrease  $C$  as much as possible. We'll constrain the size of the move so that  $\|\Delta v\| = \epsilon$  for some small fixed  $\epsilon > 0$ . In other words, we want a move that is a small step of a fixed size, and we're trying to find the movement direction which decreases  $C$  as much as possible. It can be proved that the choice of  $\Delta v$  which minimizes  $\nabla C \cdot \Delta v$  is  $\Delta v = -\eta \nabla C$ , where  $\eta = \epsilon / \|\nabla C\|$  is determined by the size constraint  $\|\Delta v\| = \epsilon$ . So gradient descent can be viewed as a way of taking small steps in the direction which does the most to immediately decrease  $C$ . Now that the gradient vector  $\nabla C$  has corresponding components  $\partial C / \partial w_k$  and  $\partial C / \partial b_l$ . Writing out the gradient descent update rule in terms of components, we have [17, 18, 19]

$$w_k \rightarrow w'_k = w_k - \eta \frac{\partial C}{\partial w_k}, \quad (2.7)$$

$$b_l \rightarrow b'_l = b_l - \eta \frac{\partial C}{\partial b_l}, \quad (2.8)$$

Now there are many challenges in training gradient-based learning. But for now, I just want to mention one problem. When the number of training inputs is very large this can take a long time and learning thus occurs slowly. An idea called stochastic gradient descent can be used to speed up learning. To make these ideas more precise, stochastic

gradient descent works by randomly picking out a small number  $m$  of randomly chosen training inputs. We'll label those random training inputs  $X_1, X_2, \dots, X_m$  and refer to them as a mini batch. Provided the sample size  $mm$  is large enough we expect that the average value of the  $\nabla C_{X_j}$  will be roughly equal to the average over all  $\nabla C_x$ , that is, [17, 19]

$$\frac{\sum_{j=1}^m}{m} \approx \frac{\sum_x \nabla C_x}{n} = \nabla C, \quad (2.9)$$

This modification helps us in reducing a good amount of computational load. Stochastic gradient descent applied to non-convex loss functions has no such convergence guarantee and is sensitive to the values of the initial parameters. For feedforward neural networks, it is important to initialize all weights to small random values. The biases may be initialized to zero or to small positive values. We will see How can we make this calculation fast by using back propagation algorithm [18].

*Back-propagation* is the essence of neural net training. It is the method of fine-tuning the weights of a neural net based on the error rate obtained in the previous epoch (i.e., iteration). Proper tuning of the weights allows you to reduce error rates and to make the model reliable by increasing its generalization [21].

Backpropagation is a short form for “backward propagation of errors”. It is a standard method of training artificial neural networks. This method helps to calculate the gradient of a loss function with respects to all the weights in the network.

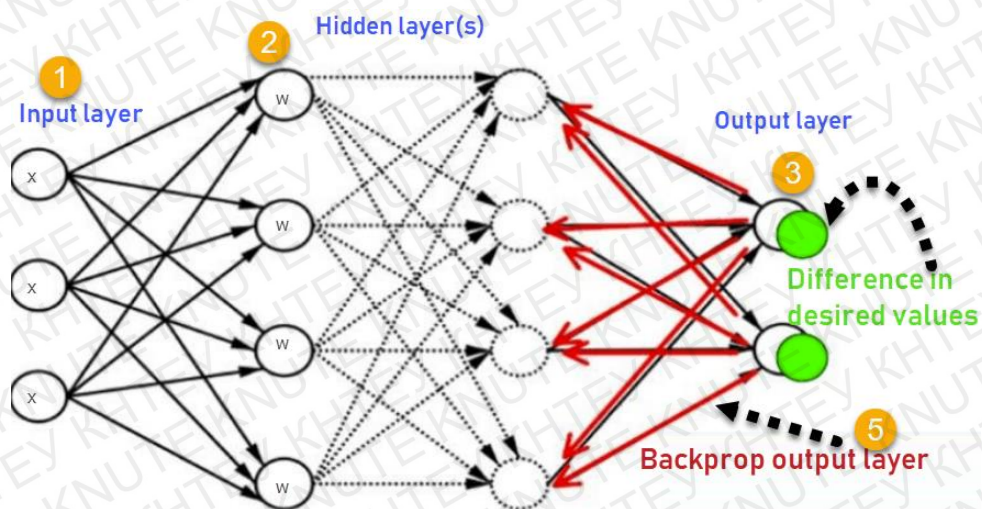


Figure 2.8. How backpropagation works

Source: designed by the author based on [21]

1. Inputs  $X$ , arrive through the preconnected path
2. Input is modeled using real weights  $W$ . The weights are usually randomly selected.
3. Calculate the output for every neuron from the input layer, to the hidden layers, to the output layer.
4. Calculate the error in the outputs
 
$$Error_B = Actual\ Output - Desired\ Output \quad (2.10)$$
5. Travel back from the output layer to the hidden layer to adjust the weights such that the error is decreased.
6. Keep repeating the process until the desired output is achieved [21].

### 2.3. Advantages of analyzing data and forecasting using neural networks

Artificial Neural Network (ANN) uses the processing of the brain as a basis to develop algorithms that can be used to model complex patterns and prediction problems.

Key advantages of neural Networks:

ANNs have some key advantages that make them most suitable for certain problems and situations:

1. ANNs have the ability to learn and model non-linear and complex relationships, which is really important because in real-life, many of the relationships between inputs and outputs are non-linear as well as complex.
2. ANNs can generalize – After learning from the initial inputs and their relationships, it can infer unseen relationships on unseen data as well, thus making the model generalize and predict on unseen data.
3. Unlike many other prediction techniques, ANN does not impose any restrictions on the input variables (like how they should be distributed). Additionally, many studies have shown that ANNs can better model heteroskedasticity i.e. data with high volatility and non-constant variance, given its ability to learn hidden relationships in the data without imposing any fixed relationships in the data. This is something very useful in financial time series forecasting (e.g. stock prices) where data volatility is very high [25].

Forecasting: Forecasting is required extensively in everyday business decisions (e.g. sales, financial allocation between products, capacity utilization), in economic and monetary policy, in finance and stock market. More often, forecasting problems are complex, for example, predicting stock prices is a complex problem with a lot of underlying factors (some known, some unseen). Traditional forecasting models throw up limitations in terms of taking into account these complex, non-linear relationships. ANNs, applied in the right way, can provide robust alternative, given its ability to model and extract unseen features and relationships. Also, unlike these traditional models, ANN doesn't impose any restriction on input and residual distributions [26].

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques. Other advantages include:

**Adaptive learning:** An ability to learn how to do tasks based on the data given for training or initial experience.

- *Self-Organization:* An ANN can create its own organization or representation of the information it receives during learning time.
- *Real Time Operation:* ANN computations may be carried out in parallel, and special hardware devices are being designed and manufactured which take advantage of this capability.
- *Fault Tolerance via Redundant Information Coding:* Partial destruction of a network leads to the corresponding degradation of performance. However, some network capabilities may be retained even with major network damage.

Neural networks are universal approximators, and they work best if the system you are using them to model has a high tolerance to error. However, they work very well for:

- capturing associations or discovering regularities within a set of patterns;
- where the volume, number of variables or diversity of the data is very great;
- the relationships between variables are vaguely understood;
- the relationships are difficult to describe adequately with conventional approaches.

The greatest strength of neural networks is their ability to accurately predict outcomes of complex problems. In accuracy tests against other approaches, neural networks are always able to score very high [24, p. 445].

### **Conclusions to the section 2**

In the following section theoretical and methodological aspects of machine learning workflow, data preparation, neural network architecture, principles of their learning and advantages of using them for prediction were analyzed. The goal of machine learning is to build systems capable of finding patterns in data, learning from it without human intervention and explicit reprogramming. To solve the price prediction problem, data scientists first must understand what data to use to train machine learning models, and that's exactly why descriptive analytics is needed.

While ML projects vary in scale and complexity requiring different data science teams, their general structure is the same. We have determined steps which should be taken to deploy a model and make prediction.

Training a model with a dataset that has a lot of missing values can drastically impact the machine learning model's quality. Some algorithms such as scikit-learn estimators assume that all values are numerical and have and hold meaningful value. So, in this section popular ways for data imputation were suggested.

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques.

## SECTION 3

### Building neural network to forecast Real Estate prices

#### 3.1 Exploratory data analysis and feature engineering

The performance of any ML model completely depends on the quality of the data that has been fed into the system. So the predictors should be chosen with the utmost care and the data should be in the proper format with reliable values. There are different steps that are involved in the data analysis process starting from the gathering of details until it is given as input to the machine learning model.

Feature engineering is the process of determining which predictor variables will contribute the most to the predictive power of a machine learning algorithm. There are two commonly used methods for making this selection – the Forward Selection Procedure starts with no variables in the model. You then iteratively add variables and test the predictive accuracy of the model until adding more variables no longer makes a positive effect. Next, the Backward Elimination Procedure begins with all the variables in the model. You proceed by removing variables and testing the predictive accuracy of the model [38].

The process of feature engineering is as much of an art as a science. Often feature engineering is a give-and-take process with exploratory data analysis to provide much needed intuition about the data. Feature engineering is when you use your knowledge about the data to select and create features that make machine learning algorithms work better [38].

One problem with machine learning is too much data. With today's big data technology, we're in a position where we can generate a large number of features. In such cases, fine-tuned feature engineering is even more important.

In the following section we are going to build model based off the dataset for different housing features and we are going to try to predict the price that a house should sell at.

We will be using data which contains house sale prices for King County, which includes Seattle. It includes homes sold between May 2014 and May 2015 [37]. Example of used data can be seen in Appendix A.

### *Feature Columns and their definition*

- 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
- view - An index from 0 to 4 of how good the view of the property was
- condition - An index from 1 to 5 on the condition of the apartment,
- 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 - Latitude
- 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

At the beginning of the project we need to import corresponding libraries and data from dataset.

Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications. Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

In [2]: df = pd.read_csv('../data/kc_house_data.csv')
```

Figure 3.1. Import libraries and data from dataset

Source: designed by the author

Next step is we need to determine if we have any missing data in the dataset, to do that we can use method `isnull()` on our data frame, this method returns true or false if some data is missing, for better visual understanding method `sum()` is used, that sums data per columns and treats False as 0 and 1 as True.



```
In [3]: df.isnull().sum()

Out[3]: id          0
        date        0
        price       0
        bedrooms    0
        bathrooms   0
        sqft_living  0
        sqft_lot     0
        floors      0
        waterfront  0
        view        0
        condition   0
        grade       0
        sqft_above  0
        sqft_basement 0
        yr_built    0
        yr_renovated 0
        zipcode     0
        lat         0
        long        0
        sqft_living15 0
        sqft_lot15  0
        dtype: int64
```

Figure 3.2. Check for missing data in dataset

Source: designed by the author

As we can see from Figure 3.2 for this particular dataset we don't have any missing values, but it's not always the case, in second section we explored how to deal with missing data.

Pandas *describe()* is used to view some basic statistical details like percentile, mean, std etc. of a data frame or a series of numeric values.

```
In [4]: df.describe().transpose()
```

```
Out[4]:
```

	count	mean	std	min	25%	50%	75%	max
id	21597.0	4.580474e+09	2.876736e+09	1.000102e+06	2.123049e+09	3.904930e+09	7.308900e+09	9.900000e+09
price	21597.0	5.402966e+05	3.673681e+05	7.800000e+04	3.220000e+05	4.500000e+05	6.450000e+05	7.700000e+06
bedrooms	21597.0	3.373200e+00	9.262989e-01	1.000000e+00	3.000000e+00	3.000000e+00	4.000000e+00	3.300000e+01
bathrooms	21597.0	2.115826e+00	7.689843e-01	5.000000e-01	1.750000e+00	2.250000e+00	2.500000e+00	8.000000e+00
sqft_living	21597.0	2.080322e+03	9.181061e+02	3.700000e+02	1.430000e+03	1.910000e+03	2.550000e+03	1.354000e+04
sqft_lot	21597.0	1.509941e+04	4.141264e+04	5.200000e+02	5.040000e+03	7.618000e+03	1.068500e+04	1.651359e+06
floors	21597.0	1.494096e+00	5.396828e-01	1.000000e+00	1.000000e+00	1.500000e+00	2.000000e+00	3.500000e+00
waterfront	21597.0	7.547345e-03	8.654900e-02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00
view	21597.0	2.342918e-01	7.663898e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.000000e+00
condition	21597.0	3.409825e+00	6.505456e-01	1.000000e+00	3.000000e+00	3.000000e+00	4.000000e+00	5.000000e+00
grade	21597.0	7.657915e+00	1.173200e+00	3.000000e+00	7.000000e+00	7.000000e+00	8.000000e+00	1.300000e+01
sqft_above	21597.0	1.788597e+03	8.277598e+02	3.700000e+02	1.190000e+03	1.560000e+03	2.210000e+03	9.410000e+03
sqft_basement	21597.0	2.917250e+02	4.426678e+02	0.000000e+00	0.000000e+00	0.000000e+00	5.600000e+02	4.820000e+03
yr_built	21597.0	1.971000e+03	2.937523e+01	1.900000e+03	1.951000e+03	1.975000e+03	1.997000e+03	2.015000e+03
yr_renovated	21597.0	8.446479e+01	4.018214e+02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.015000e+03
zipcode	21597.0	9.807795e+04	5.351307e+01	9.800100e+04	9.803300e+04	9.806500e+04	9.811800e+04	9.819900e+04
lat	21597.0	4.756009e+01	1.385518e-01	4.715590e+01	4.747110e+01	4.757180e+01	4.767800e+01	4.777760e+01
long	21597.0	-1.222140e+02	1.407235e-01	-1.225190e+02	-1.223280e+02	-1.222310e+02	-1.221250e+02	-1.213150e+02
sqft_living15	21597.0	1.986620e+03	6.852305e+02	3.990000e+02	1.490000e+03	1.840000e+03	2.360000e+03	6.210000e+03
sqft_lot15	21597.0	1.275828e+04	2.727444e+04	6.510000e+02	5.100000e+03	7.620000e+03	1.008300e+04	8.712000e+05

Figure 3.3. Statistical analysis of the dataset

Source: designed by the author

It's a little bit hard to read this table and thoroughly understand it. Let's start describing it through visualization using matplotlib. For continuous labels we can do distribution of the actual label(price) as on Figure 3.4. Continuous Data represents measurements and therefore their values can't be counted but they can be measured.

```
In [5]: plt.figure(figsize=(12,8))  
sns.distplot(df['price'])  
  
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x26e56791108>
```

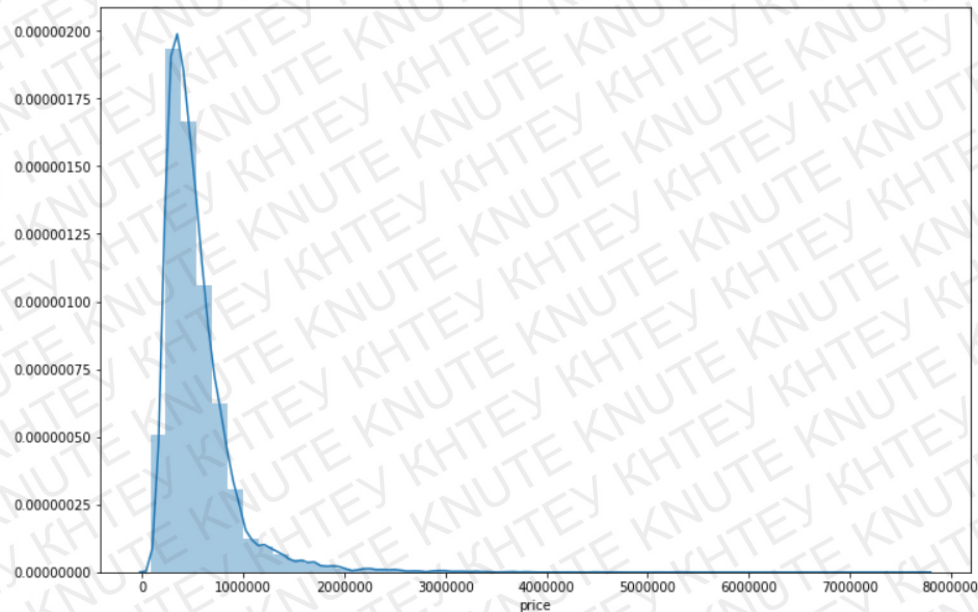


Figure 3.4. Distribution plot (histogram)

Source: designed by the author

As a result, we can see that most of our houses are falling somewhere between zero and 1.5 million dollars. We also have extreme outliers for really expensive houses and it may actually make sense to drop those outliers in our analysis if they are just a few points that are very extreme, so we can essentially build a model that realistically predicts the price of the house if it's intended value somewhere between 0 and 2 million dollars. Since it's really not that many houses on the market that are that expensive it may not be really useful to actually have our model train on these extreme outliers.

Now we can go ahead and do similar analysis of different features. For categorical ones such as numbers of bedrooms.

```
In [6]: sns.countplot(df['bedrooms'])
```

```
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x26e56900ec8>
```

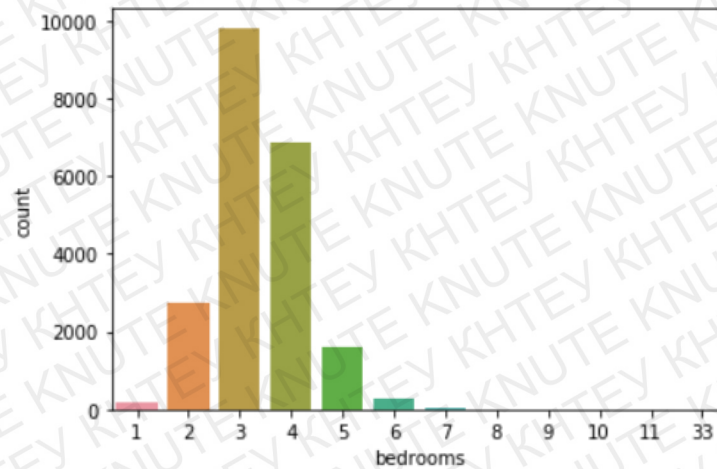


Figure 3.5. Countplot for bedrooms

Source: designed by the author

Here we can see what actually looks like similar distribution where the vast majority of all houses have somewhere between two to five bedrooms and there is huge mansion that has 33 bedrooms. What is also useful is comparing your label to some sort of feature that you think has a high correlation.

We can explore highly correlated features using scatterplot.

```
In [7]: plt.figure(figsize=(12,8))
sns.scatterplot(x='price',y='sqft_living',data=df)

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x26e548ed688>
```

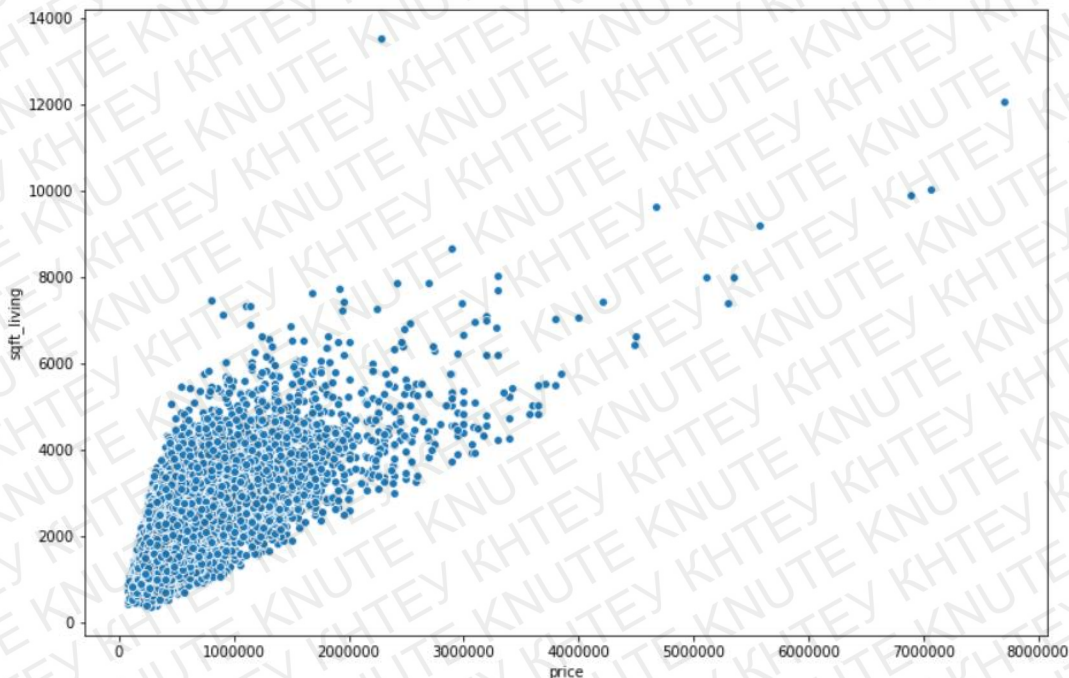


Figure 3.6. Scatterplot to compare correlation between price to sqft\_living space

Source: designed by the author

We can see here a very strong linear relationship. It is recommended checking out correlations between different features and actual label, which is price. Bedrooms also have some positive correlation. We can do box plot to see the distributions.

```
In [8]: sns.boxplot(x='bedrooms',y='price',data=df)

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x26e569ae388>
```

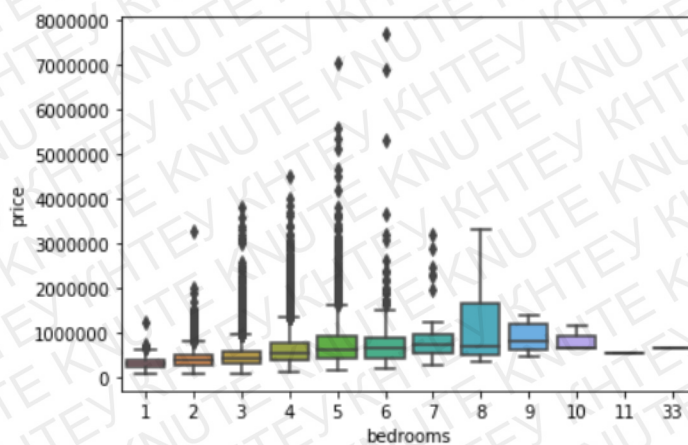


Figure 3.7. Box blot to compare correlation between price and number of bedrooms

Source: designed by the author

What this is showing us is distribution of prices per bedrooms. We can see quite a bit of variation in bedrooms ranging between 3 and 7.

### Geographical properties

In our dataset we have features such as latitude and longitude and it may be useful to actually explore this by plotting it out using simple scatterplot. We are not going to focus on trying to plot these points on a real world map; instead, we can gain a lot of information with a little bit of cursory knowledge of what King County looks like combined with a simple scatterplot call.

So, let's first see the distribution of prices per latitude vs longitude.

```
In [9]: plt.figure(figsize=(12,8))
sns.scatterplot(x='price',y='long',data=df)
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x26e570d1748>
```

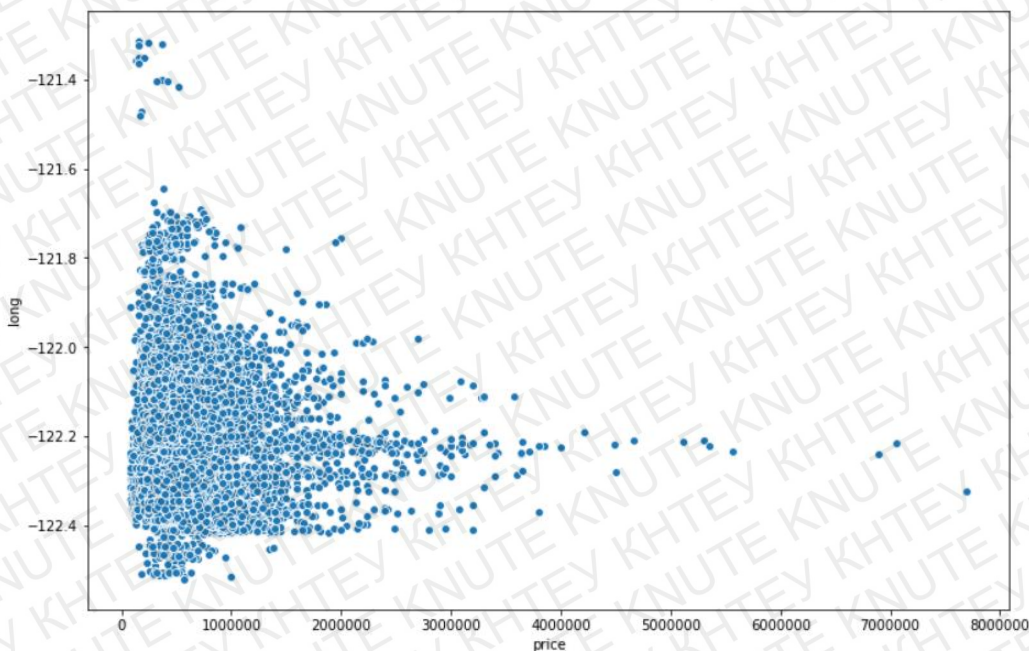


Figure 3.8. Scatterplot to compare correlation between price to longitude

Source: designed by the author

It looks like there tends to be some sort of price distribution at a certain longitude, at longitude -122.2 looks like an expensive housing area, we can see the distribution quite clearly here. Now we can repeat it for latitude.

```
In [10]: plt.figure(figsize=(12,8))  
sns.scatterplot(x='price',y='lat',data=df)  
  
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x26e57184bc8>
```

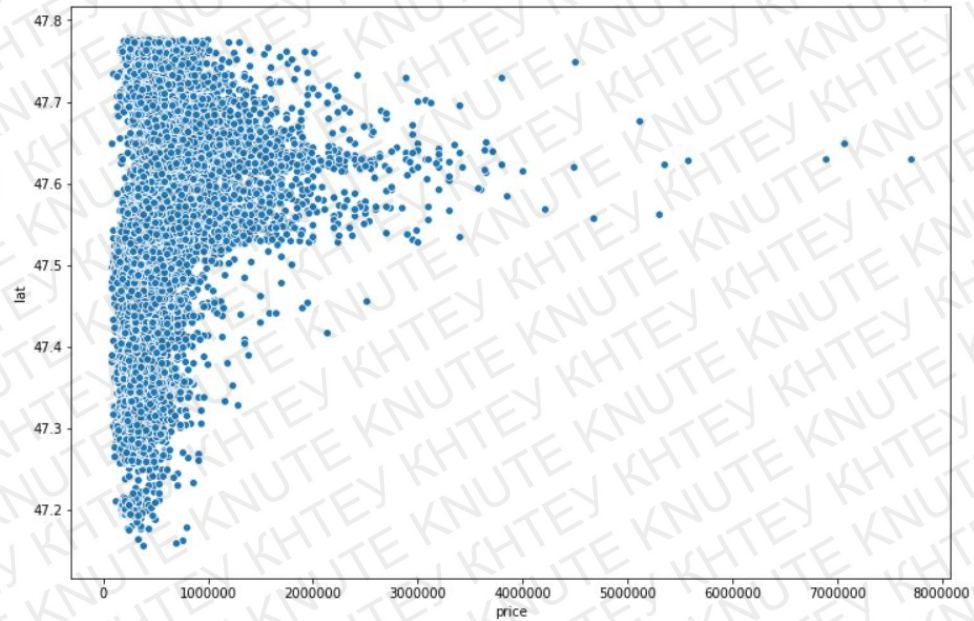


Figure 3.8. Scatterplot to compare correlation between price to latitude

Source: designed by the author

As we can see the same behavior seems to pop up, it also seems that at particular latitude (47.7, 47.6) there is expensive housing area. What this is telling us is that at a certion combination of latitude and longitude that tends to be an expensive area. Let's plot out latitude versus longitude and plot out all these points and latter we can affect their hue.

```
In [11]: plt.figure(figsize=(12,8))
sns.scatterplot(x='long',y='lat',data=df,hue='price')
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x26e571e78c8>
```

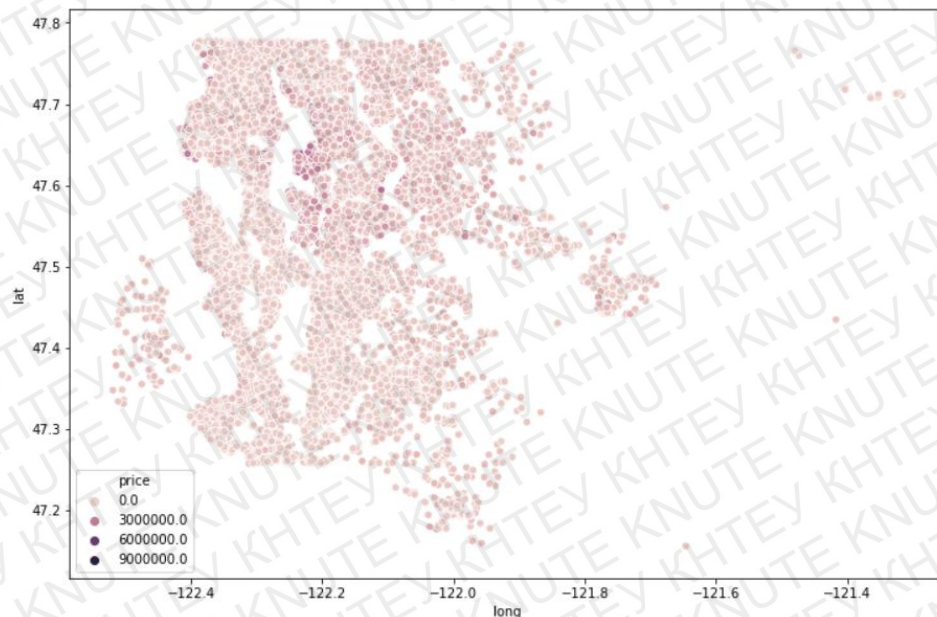


Figure 3.9. Scatterplot latitude and longitude

Source: designed by the author

If we compare this to the actual map of King County, we can see that more or less they match. And what are we going to do now is we are going to start editing this to see if we can actually hone in on this expensive housing area. We can begin to see a little bit here of a darker area and it looks like It's mathing up with our original estimates of the expensive longitude, however we are not getting gradient color as we would like.



```
In [14]: non_top_1_perc = df.sort_values('price', ascending=False).iloc[216:]
```

```
In [15]: plt.figure(figsize=(12,8))
sns.scatterplot(x='long',y='lat',
               data=non_top_1_perc,hue='price',
               palette='RdYlGn',edgecolor=None,alpha=0.2)
```

```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x26e578a5388>
```

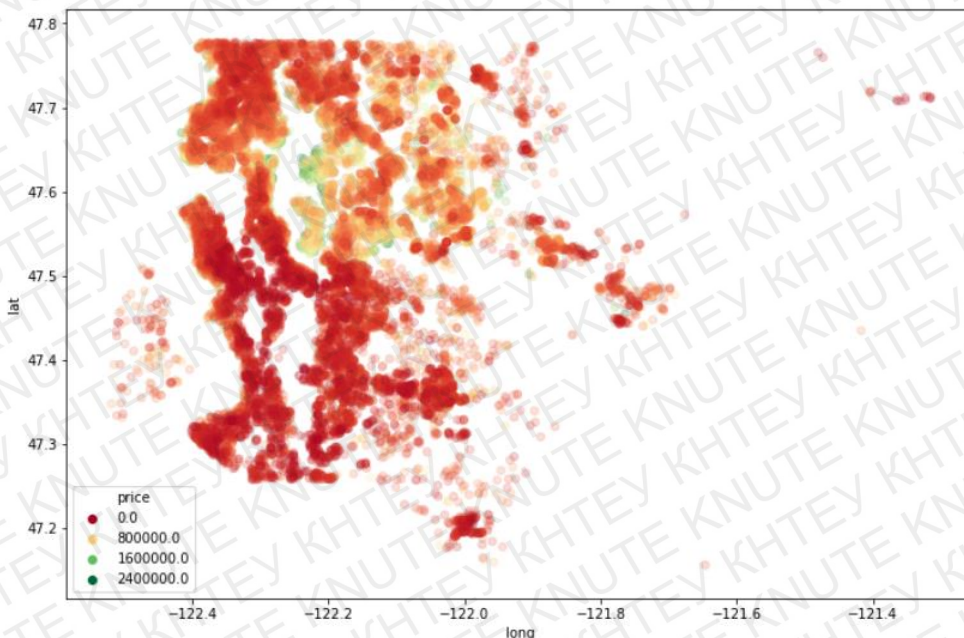


Figure 3.10. Scatterplot latitude and longitude with gradient color

Source: designed by the author

Now we can definitely see a lot clearer color distribution. This plot showing us where the expensive parts of King County is.

Next, we convert date columns from dataset into datetime object on line 21, that means that now we can extract information such as the month or year automatically. Now we can begin feature engineering off of this.

```
In [21]: df['date'] = pd.to_datetime(df['date'])
In [22]: df['month'] = df['date'].apply(lambda date:date.month)
In [23]: df['year'] = df['date'].apply(lambda date:date.year)
In [24]: sns.boxplot(x='year',y='price',data=df)
Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x26e59114848>
```

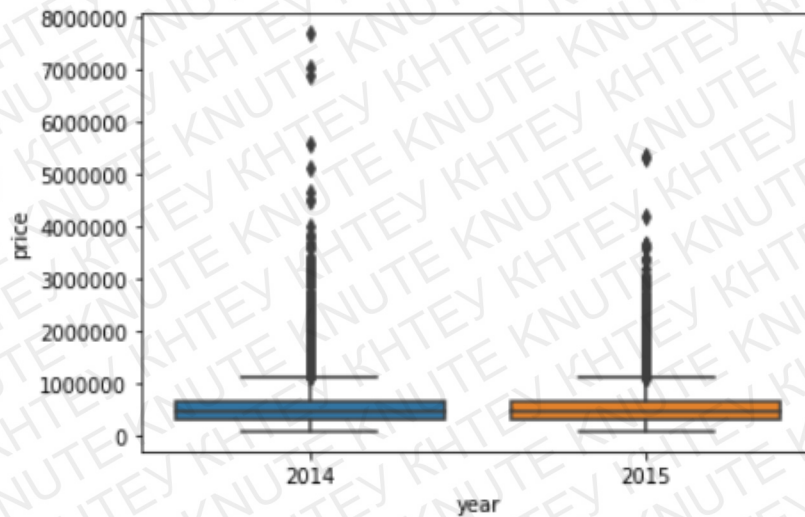


Figure 3.11. Boxplot to see correlation between price and year

Source: designed by the author

Using lamda function we extract year and month from date object, this is essentially feature engineering because these features were technically hidden inside of the string data, and now we're creating new columns to try to extract or engineer more information off original features.

Now we can do this for month as well.

```
In [25]: sns.boxplot(x='month',y='price',data=df)
```

```
Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x26e593b1f48>
```

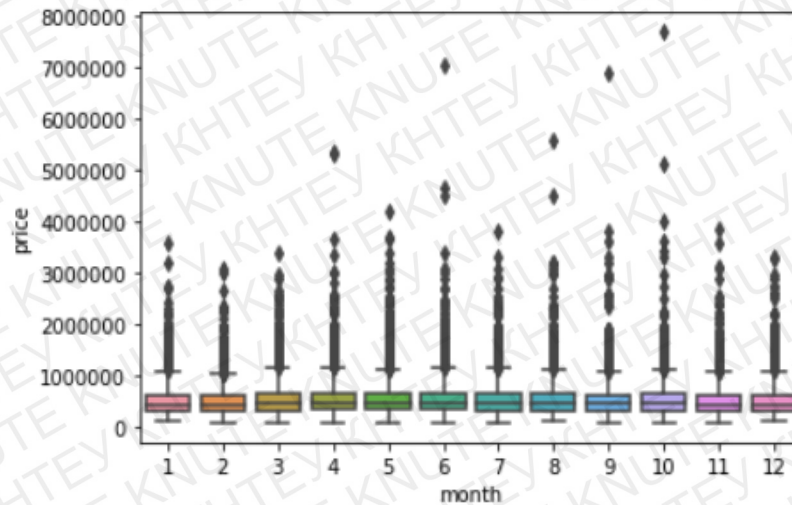


Figure 3.12. Boxplot to see correlation between price and month.

Source: designed by the author

Now it's hard to tell just from this plot whether or not there is any significant distribution differences between the month you are going to sell this house at. What might help is to see the numbers themselves. This will allow us to see the numbers and see if there is some significant difference between the months.

```
In [26]: df.groupby('month').mean()['price'].plot()
```

```
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x26e593bbc48>
```

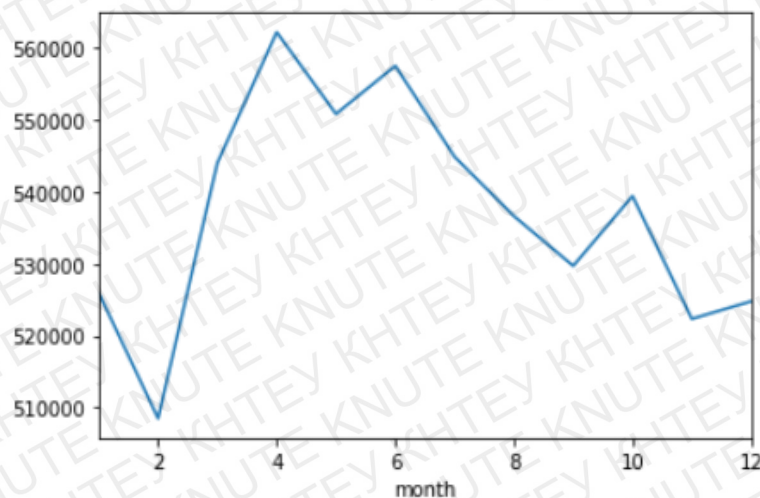


Figure 3.13. Groupby to see correlation between price and months.

Source: designed by the author

As we can see from the plot, not a huge range but it's looks like there is some behavior difference there in the month themselves. We can do similar thing for the year.

```
In [27]: df.groupby('year').mean()['price'].plot()
```

```
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x26e5959ed48>
```

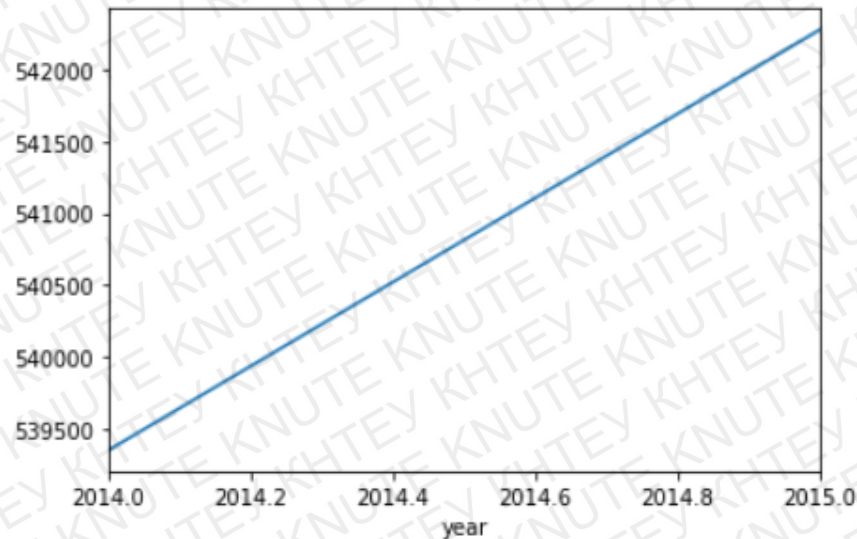


Figure 3.13. Groupby to see correlation between price and year.

Source: designed by the author

This plot definitely makes sense because if you look back at the King County sales they are just increasing in price as time goes on.

### 3.2 Creating and training the model

We have already taken a look at the features then a little bit of feature engineering and even dropped one feature. Now it's time to scale along for train test split and then create a model and train the model.

The nex step is to separate our features from our label(price). After separation features from our label it's time to do a train test split.

```
In [35]: X = df.drop('price',axis=1)
         y = df['price']
```

```
In [36]: from sklearn.model_selection import train_test_split
```

```
In [37]: X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=101)
```

Figure 3.14. Separate features from label(price)

Source: designed by the author

Now we want to perform a scaling, we want to perform a scaling post split, that way we only fit to the training set to prevent data leakage from the test set.

```
In [38]: from sklearn.preprocessing import MinMaxScaler
```

```
In [39]: scaler = MinMaxScaler()
```

```
In [40]: X_train = scaler.fit_transform(X_train)
```

```
In [41]: X_test = scaler.transform(X_test)
```

```
In [42]: X_train.shape
```

```
Out[42]: (15117, 19)
```

```
In [43]: X_test.shape
```

```
Out[43]: (6480, 19)
```

Figure 3.15. Performing post split scaling

Source: designed by the author

Coming up next, we are going to do is create the model. We create sequential model and typically what we do is we try to base the number of neurons or units in our layers from the size the actual feature data. From the X\_train.shape it looks like we have 19 incoming features and that's probably a good range to then have 19 neurons in our layer.

```
In [44]: from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Activation
from tensorflow.keras.optimizers import Adam
```

```
In [45]: model = Sequential()
```

```
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(19,activation='relu'))
model.add(Dense(1))
```

```
model.compile(optimizer='adam',loss='mse')
```

Figure 3.16. Creating model

Source: designed by the author

So, we added one final layer and this layer is going to have one neuron as it's output since that's going to be directly outputting our predicted price. And then we are going to compile this model.

Now it's time to train a model.

```
In [46]: model.fit(x=X_train,y=y_train.values,
                 validation_data=(X_test,y_test.values),
                 batch_size=128,epochs=400)
```

```
Train on 15117 samples, validate on 6480 samples
Epoch 1/400
15117/15117 [=====] - 1s 74us/sample - loss: 430228839929.3955 - val_loss: 418844493758.26
17
Epoch 2/400
15117/15117 [=====] - 0s 21us/sample - loss: 428253102583.9730 - val_loss: 411996069786.86
42
Epoch 3/400
15117/15117 [=====] - 0s 21us/sample - loss: 401401261499.3472 - val_loss: 351554389222.08
39
Epoch 4/400
15117/15117 [=====] - 0s 22us/sample - loss: 281418027052.1653 - val_loss: 178278463958.28
15
Epoch 5/400
15117/15117 [=====] - 0s 21us/sample - loss: 126127754575.5078 - val_loss: 95946680446.419
8
Epoch 6/400
15117/15117 [=====] - 0s 21us/sample - loss: 97670204498.9794 - val_loss: 93708013133.1160
Epoch 7/400
```

Figure 3.17. Training model

Source: designed by the author

We are passing validation data, and what that means is after each epoch of training on the training data will quickly run the test data and check our loss on the test data. So that way we can keep tracking of how well we are performing not just on our training data but also on our test data.

Now it's time to explore and evaluate on not just our test data but also being able to predict the price of new house given it's features.

```
In [47]: losses = pd.DataFrame(model.history.history)
```

```
In [48]: losses.plot()
```

```
Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x2707c650b88>
```

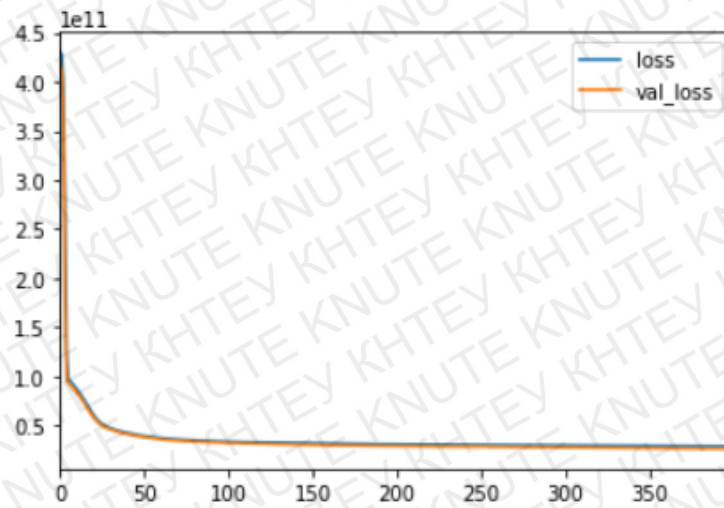


Figure 3.18. Plot of losses after model training

Source: designed by the author

Now we can directly compare the loss on training versus the lost on test and validation or order to see if we are over fitting to the training data on our model.

This is exactly the kind of signal we want where there is decrease in both the training loss the validation loss. That is indicator that we could continue training without over fitting to our training data.

```
In [51]: predictions = model.predict(X_test)

In [52]: mean_absolute_error(y_test, predictions)

Out[52]: 101666.74811137635

In [53]: np.sqrt(mean_squared_error(y_test, predictions))

Out[53]: 162549.26352210157

In [54]: explained_variance_score(y_test, predictions)

Out[54]: 0.8009472831860807

In [55]: df['price'].mean()

Out[55]: 540296.5735055795

In [56]: df['price'].median()

Out[56]: 450000.0
```

Figure 3.19. Test versus predictions

Source: designed by the author

Looking on our mean absolute error, we are off by about one hundred thousand dollars.

We have to take into account the actual data frame itself, we have to take into account our price column off the original data frame. We can also compare our prediction and we can plot them against a perfect fit.



```
In [57]: plt.scatter(y_test,predictions)
plt.plot(y_test,y_test,'r')
```

```
Out[57]: [<matplotlib.lines.Line2D at 0x2707a07fe08>]
```

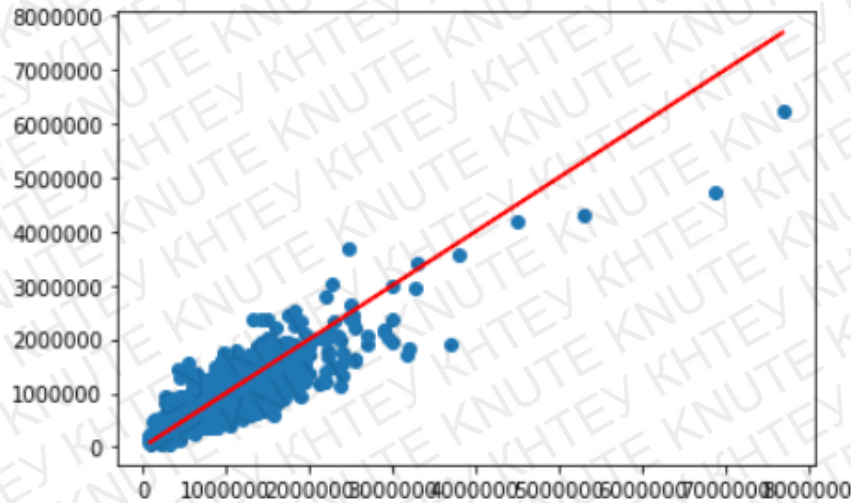


Figure 3.20. Plot compare prediction versus perfect fit

Source: designed by the author

The red line represents perfect prediction line. We can notice that we are basically being punished here by expensive houses. These really expensive houses were not good at predicting that price, but we are good at prediction house prices between 0 and 2 million dollars.

### 3.3. Model usage and evaluation

So finally let's show how model would to predict on a brand new house. There are only the features of a new house on market. So let's say a new house is coming onto the market and someone wants to sell and we know these various features of the house based on which we are going to predict price.

We drop price and now have all these features for a new house coming onto the market. The next step is to scale single\_house data.

```

In [81]: single_house = df.drop('price',axis=1).iloc[0]

In [86]: single_house = scaler.transform(single_house.values.reshape(-1, 19))

In [87]: single_house
Out[87]: array([[0.22222222, 0.08      , 0.11392405, 0.00433777, 0.
0.          , 0.          , 0.5       , 0.375     , 0.11654676,
0.          , 0.47826087, 0.          , 0.56914119, 0.21760797,
0.16510319, 0.00574235, 0.81818182, 0.          ]])

In [89]: model.predict(single_house)
Out[89]: array([[283862.12]], dtype=float32)

```

Figure 3.21. House price prediction

Source: designed by the author

After training and running the model, predicted price for the house is 283852, as you can see on figure 3.22 original price of the house was 271900, which tells that our model for house price prediction is pretty accurate. The full progress of development can be seen in Appendix B.

```

Out[90]: price                271900.0000
bedrooms                   3.0000
bathrooms                   1.0000
sqft_living                 1180.0000
sqft_lot                    5650.0000
floors                       1.0000
waterfront                   0.0000
view                         0.0000
condition                    3.0000
grade                         7.0000
sqft_above                  1180.0000
sqft_basement                 0.0000
yr_built                     1955.0000
yr_renovated                  0.0000
lat                           47.5112
long                         -122.2570
sqft_living15                1340.0000
sqft_lot15                    5650.0000
month                          10.0000
year                          2014.0000
Name: 0, dtype: float64

```

Figure 3.22 Original house features

Source: designed by the author

What may help to make prediction more accurate is to retrain model just on that bottom ninety nine percent of houses, if we come up to a situation where our sale price is over three million dollars we'll only refit to that bottom 99 percent. It really depends on the context and what questions are trying to be answered and what problems are trying to be solved. It looks like we are kind of overshooting price a little bit and again that may be an issue when we're trying to fit to these extreme values. next step to undertake would be to retrain model by dropping out the top 1 or 2 percent of values and see if that can reduce model means squared error on the data set.

### **Conclusions to the section 3**

In the first part of the section we have been able to do a little bit of fixed engineering, the process of feature engineering is as much of an art as a science. Often feature engineering is a give-and-take process with exploratory data analysis to provide much needed intuition about the data. Feature engineering is when you use your knowledge about the data to select and create features that make machine learning algorithms work better. We have seen how different features can correlate with our label, data that we want to predict, that often in datasets there is data which can disturb accurate prediction of the model. We have done a little bit of exploratory data analysis.

In the second part of the section we created and trained the model, with help of passing validation data, after each epoch of training on the training data will quickly run the test data and check our loss on the test data. So that way we can keep tracking of how well we are performing not just on our training data but also on our test data. We directly compared the loss on training versus the loss on test and validation in order to see if we are over fitting to the training data on our model which resulted in exactly the kind of signal we want where there is decrease in both the training loss the validation loss. That is indicator that we could continue training without over fitting to our training data. After running our model to predict price of the house based on its features got back a relatively reasonable model.

## CONCLUSIONS

In the graduation qualification work theoretical, methodological aspects of real estate price forecasting using neural network were analyzed and developed practical solution for price prediction. Theoretical aspects of real estate were revealed. After analyzing scientific literature, we have defined economic characteristics of real estate that influence its value as an investment. The results of the study allow us to draw the following conclusions:

Estimating the value of real estate is necessary for a variety of endeavors, including financing, sales listing, investment analysis, property insurance, and taxation. But for most people, determining the asking or purchase price of a piece of real property is the most useful application of real estate valuation. We have determined 3 traditional methods to run a valuation on property: comparable sales approach, Income approach, cost approach

Accurate real estate valuation is important to mortgage lenders, investors, insurers and buyers, and sellers of real property. While appraisals are generally performed by skilled professionals, anyone involved in a real transaction can benefit from gaining a basic understanding of the different methods of real estate valuation.

After analyzing aspects of machine learning workflow, data preparation, neural network architecture, principles of their learning and advantages of using them for prediction were determined that the goal of machine learning is to build systems capable of finding patterns in data, learning from it without human intervention and explicit reprogramming. To solve the price prediction problem, data scientists first must understand what data to use to train machine learning models, and that's exactly why descriptive analytics is needed.

While ML projects vary in scale and complexity requiring different data science teams, their general structure is the same. We have determined steps which should be taken to deploy a model and make prediction.

Training a model with a dataset that has a lot of missing values can drastically impact the machine learning model's quality. Some algorithms such as scikit-learn

estimators assume that all values are numerical and have and hold meaningful value. So, in this section popular ways for data imputation were suggested.

Neural networks, with their remarkable ability to derive meaning from complicated or imprecise data, can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques.

In order to find correlation between different feature on our price, we have been able to do fixed engineering and exploratory data analysis. The process of feature engineering is as much of an art as a science. Often feature engineering is a give-and-take process with exploratory data analysis to provide much needed intuition about the data. Feature engineering is when you use your knowledge about the data to select and create features that make machine learning algorithms work better. We have seen how different features can corelete with our label, data that we want to predict, that offen in datasets there is data which can disturb accurate prediction of the model.

To use prepared data after feature engineering we created and trained the model, with help of passing validation data, after each epoch of training on the training data will quickly run the test data and check our loss on the test data. So that way we can keep tracking of how well we are performing not just on our training data but also on our test data. We directly compared the loss on training versus the lost on test and validation or order to see if we are over fitting to the training data on our model which resulted in exactly the kind of signal we want where there is decrease in both the training loss the validation loss. That is indicator that we could continue training without over fitting to our training data. After running our model to predict price of the house based on its features got back a relatively reasonable model. As a result we received more general, robust and accurate method of price forecasting which could be used by investment companies to predict real estate price and make good investment, personal benefit to predict real estate price based on different factors to buy house at the right price.

## REFERENCES

1. Chen J. Real Estate Investing. Investopedia. September 2, 2020. [Electronic resource]. Access mode: <https://www.investopedia.com/terms/r/realestate.asp> (date of access: 21.09.2020).
2. Passmasters. What are the economic characteristics of land? (Blog) [Electronic resource]. Access mode: <https://passmasters.com/article/property-ownership/economic-characteristics-of-land/> (date of access: 21.09.2020).
3. Amadeo K. Real Estate, What It Is and How It Works. The Balance. February 24, 2020. [Electronic resource]. Access mode: <https://www.thebalance.com/real-estate-what-it-is-and-how-it-works-3305882> (date of access: 21.09.2020).
4. Ball M. 4 Property Valuation Methods for Real Estate Investors. LoftyAI. May 12, 2020. [Electronic resource]. Access mode: <https://www.lofty.ai/blog/property-valuation-methods-real-estate-investors> (date of access: 13.08.2020).
5. Folger J. What You Should Know About Real Estate Valuation. March 3, 2020. [Electronic resource]. Access mode: <https://www.investopedia.com/articles/realestate/12/real-estate-valuation.asp> (date of access: 13.08.2020).
6. Obaidullah J. Real Estate Valuation. XPLAIND. June 10, 2019. [Electronic resource]. Access mode: <https://xplained.com/726953/real-estate-valuation> (date of access: 13.08.2020).
7. Allabadi A. 3 Main Property Valuation Methods for Real Estate Investors. Mashvisor. April 15, 2018. [Electronic resource]. Access mode: <https://www.mashvisor.com/blog/property-valuation-methods-real-estate-investors/> (date of access: 13.08.2020).
8. International Monetary Fund. Real Estate Price Indices // Financial Soundness Indicators: Compilation Guide. 2016. 101–107 p. [Electronic resource]. Access mode: <https://www.imf.org/external/pubs/ft/fsi/guide/2006/pdf/chp9.pdf> (date of access: 13.08.2020).

9. NYS Department of Taxation & Finance Office of Real Property Tax Services. How to Estimate the Market Value of Your Home. Albany, New York. 2018. [Electronic resource]. Access mode: [https://www.tax.ny.gov/pubs\\_and\\_bulls/orpts/mv\\_estimates.htm](https://www.tax.ny.gov/pubs_and_bulls/orpts/mv_estimates.htm) (date of access: 13.08.2020).
10. Altexsoft. Price Forecasting: Applying Machine Learning Approaches. February 26, 2019. [Electronic resource]. Access mode: <https://www.altexsoft.com/blog/business/price-forecasting-machine-learning-based-approaches-applied-to-electricity-flights-hotels-real-estate-and-stock-pricing/> (date of access: 21.05.2020).
11. Bard W. 6 Different Ways to Compensate for Missing Values In a Dataset. // Towards Data Science. January 5, 2019. [Electronic resource]. Access mode: <https://towardsdatascience.com/6-different-ways-to-compensate-for-missing-values-data-imputation-with-examples-6022d9ca0779> (date of access: 21.05.2020).
12. Altexsoft. Machine Learning Project Structure: Stages, Roles, and Tools. February 22, 2018. [Electronic resource]. Access mode: <https://www.altexsoft.com/blog/datascience/machine-learning-project-structure-stages-roles-and-tools/> (date of access: 21.05.2020).
13. Pant A. Workflow of a Machine Learning project. // Towards Data Science. January 11, 2019. [Electronic resource]. Access mode: <https://towardsdatascience.com/workflow-of-a-machine-learning-project-ec1dba419b94> (date of access: 21.05.2020).
14. Bilogurov A. Simple techniques for missing data imputation. April 28, 2018. [Electronic resource]. Access mode: <https://www.kaggle.com/residentmario/simple-techniques-for-missing-data-imputation> (date of access: 21.05.2020)
15. Sauro J. 7 Ways to handle missing data. June 2, 2015. [Electronic resource]. Access mode: <https://measuringu.com/handle-missing-data/> (date of access 21.05.2020)

16. Grace K. Seven ways to make up data. February 25, 2019.  
[Electronic resource]. Access mode: <https://www.theanalysisfactor.com/seven-ways-to-make-up-data-common-methods-to-imputing-missing-data/>
17. Gupta T. Deep Learning: Feedforward Neural Network. // Towards Data Science. January 5, 2017. [Electronic resource]. Access mode: <https://towardsdatascience.com/deep-learning-feedforward-neural-network-26a6705dbdc7> (date of access: 21.05.2020).
18. Wikipedia. Multilayer perceptron. [Electronic resource]. Access mode: [https://en.wikipedia.org/wiki/Multilayer\\_perceptron#:~:text=A%20multilayer%20perceptron%20\(MLP\)%20is,activation\)%%3B%20see%20%C2%A7%20Terminology](https://en.wikipedia.org/wiki/Multilayer_perceptron#:~:text=A%20multilayer%20perceptron%20(MLP)%20is,activation)%%3B%20see%20%C2%A7%20Terminology) (date of access: 21.05.2020).
19. Kumamoto University. Artificial Neural Network (ANN). [Electronic resource]. Access mode: <http://www.cs.kumamoto-u.ac.jp/epslab/ICinPS/Lecture-2.pdf>
20. Wikipedia. Backpropagation. [Electronic resource]. Access mode: <https://en.wikipedia.org/wiki/Backpropagation> (date of access: 21.05.2020).
21. Guru 99. Back Propagation Neural Network: Explained With Simple Example. [Electronic resource]. Access mode: <https://www.guru99.com/backpropagation-neural-network.html#2> (date of access: 21.05.2020).
22. Google Cloud. Guide: Machine learning workflow. 2020. [Electronic resource]. Access mode: <https://cloud.google.com/ai-platform/docs/ml-solutions-overview> (date of access: 21.05.2020).
23. Hagerty P. Establishing a Machine Learning Workflow // The DownLinQ. September 3, 2016. [Electronic resource]. Access mode: <https://medium.com/the-downlinq/establishing-a-machine-learning-workflow-530628cfe67> (date of access: 21.05.2020).
24. Epure D.E. // Ovidus University Press. Economic Scientific Series. 2013. № 1-2 (59). 32–37 p. [Electronic resource]. Access mode: <http://stec.univ-ovidius.ro/html/anale/ENG/cuprins%20rezumate/volum2013p1.pdf> (date of access: 21.05.2020).



25. Mahanta J. Introduction to Neural Networks, Advantages and Applications. // Towards Data Science. July 10, 2017. [Electronic resource]. Access mode: <https://towardsdatascience.com/introduction-to-neural-networks-advantages-and-applications-96851bd1a207> (date of access: 21.05.2020).
26. Thomas M. Neural Networks: Advantages and Applications. // Marktechpost. April 18, 2019. [Electronic resource]. Access mode: <https://www.marktechpost.com/2019/04/18/introduction-to-neural-networks-advantages-and-applications/> (date of access: 21.05.2020).
27. Sarkar S. Predicting House prices using Classical Machine Learning and Deep Learning techniques. // Analytics Vidhya. September 2, 2019. [Electronic resource]. Access mode: <https://medium.com/analytics-vidhya/predicting-house-prices-using-classical-machine-learning-and-deep-learning-techniques-ad4e55945e2d>
28. Ravikumar A.S. Real Estate Price Prediction Using Machine Learning. National College of Ireland. 2017. [Electronic resource]. Access mode: <http://norma.ncirl.ie/3096/1/aswinsivamravikumar.pdf> (date of access: 10.07.2020).
29. Pontus N. Prediction of residential real estate using neural network [Electronic resource]. Access mode: <https://www.diva-portal.org/smash/get/diva2:1304961/FULLTEXT01.pdf> (date of access: 10.07.2020).
30. Nilsson P. Prediction of residential real estate selling prices using neural networks. KTH Royal Institute of Technology, School of Electrical Engineering and Computer Science. Stockholm, Sweden. 2019. [Electronic resource]. Access mode: [https://www.researchgate.net/publication/282158904\\_Artificial\\_neural\\_networks\\_for\\_predicting\\_real\\_estate\\_prices](https://www.researchgate.net/publication/282158904_Artificial_neural_networks_for_predicting_real_estate_prices) (date of access: 10.07.2020).
31. Wang L. Predicting Public Housing Prices Using Delayed Neural Networks. IEEE Region 10 Conference (TENCON) – Proceedings of the International Conference. 2016. [Electronic resource]. Access mode:

[https://www.researchgate.net/publication/309443696\\_Housing\\_price\\_prediction\\_using\\_neural\\_networks](https://www.researchgate.net/publication/309443696_Housing_price_prediction_using_neural_networks) (date of access: 10.07.2020).

32. Tabales N. Artificial neural networks for predicting real estate prices. // Leibniz Information Centre for Economics. 2013. [Electronic resource]. Access mode: <https://www.econstor.eu/bitstream/10419/113851/1/756619068.pdf> (date of access: 10.07.2020).
33. Start-Tech Academy. Neural Networks in Python: Deep Learning for Beginners. Udemy course. [Electronic resource]. Access mode: <https://www.udemy.com/course/neural-network-understanding-and-building-an-ann-in-python/> (date of access: 10.07.2020).
34. Portilla J. Complete Tensorflow 2 and Keras Deep Learning Bootcamp. Udemy course. [Electronic resource]. Access mode: <https://www.udemy.com/course/complete-tensorflow-2-and-keras-deep-learning-bootcamp/> (date of access: 10.07.2020).
35. Muellauer P. Complete 2020 Data Science & Machine Learning Bootcamp. Udemy course. [Electronic resource]. Access mode: <https://www.udemy.com/course/python-data-science-machine-learning-bootcamp/> (date of access: 10.07.2020).
36. Katanforoosh K. Neural Networks and Deep Learning. Udemy course. [Electronic resource]. Access mode: <https://www.coursera.org/learn/neural-networks-deep-learning?specialization=deep-learning> (date of access: 10.07.2020).
37. Dataset. House Sales in King Country, USA [Electronic resource]. Access mode: <https://www.kaggle.com/harlfoxem/housesalesprediction> (date of access: 10.07.2020).
38. Daniel G. Data Munging, Exploratory Data Analysis, and Feature Engineering [Electronic resource]. Access mode: <https://insidebigdata.com/2014/06/05/data-munging-exploratory-data-analysis-feature-engineering/>

## APPENDICES

## Appendix A

## Example of data from dataset

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
1	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_baser	yr_built	yr_renova	zipcode	lat	long	sqft_living	sqft_lot15
2	7.13E+09	20141013	221900	3	1	1180	5650	1	0	0	3	7	1180	0	1955	0	98178	47.5112	-122.257	1340	5650
3	6.41E+09	20141209	538000	3	2.25	2570	7242	2	0	0	3	7	2170	400	1951	1991	98125	47.721	-122.319	1690	7639
4	5.63E+09	20150225	180000	2	1	770	10000	1	0	0	3	6	770	0	1933	0	98028	47.7379	-122.233	2720	8062
5	2.49E+09	20141209	604000	4	3	1960	5000	1	0	0	5	7	1050	910	1965	0	98136	47.5208	-122.393	1360	5000
6	1.95E+09	20150218	510000	3	2	1680	8080	1	0	0	3	8	1680	0	1987	0	98074	47.6168	-122.045	1800	7503
7	7.24E+09	20140512	1.23E+06	4	4.5	5420	101930	1	0	0	3	11	3890	1530	2001	0	98053	47.6561	-122.005	4760	101930
8	1.32E+09	20140627	257500	3	2.25	1715	6819	2	0	0	3	7	1715	0	1995	0	98003	47.3097	-122.327	2238	6819
9	2.01E+09	20150115	291850	3	1.5	1060	9711	1	0	0	3	7	1060	0	1963	0	98198	47.4095	-122.315	1650	9711
10	2.41E+09	20150415	229500	3	1	1780	7470	1	0	0	3	7	1050	730	1960	0	98146	47.5123	-122.337	1780	8113
11	3.79E+09	20150312	323000	3	2.5	1890	6560	2	0	0	3	7	1890	0	2003	0	98038	47.3684	-122.031	2390	7570
12	1.74E+09	20150403	662500	3	2.5	3560	9796	1	0	0	3	8	1860	1700	1965	0	98007	47.6007	-122.145	2210	8925
13	9.21E+09	20140527	468000	2	1	1160	6000	1	0	0	4	7	860	300	1942	0	98115	47.69	-122.292	1330	6000
14	1.14E+08	20140528	310000	3	1	1430	19901	1.5	0	0	4	7	1430	0	1927	0	98028	47.7558	-122.229	1780	12697
15	6.05E+09	20141007	400000	3	1.75	1370	9680	1	0	0	4	7	1370	0	1977	0	98074	47.6127	-122.045	1370	10208
16	1.18E+09	20150312	530000	5	2	1810	4850	1.5	0	0	3	7	1810	0	1900	0	98107	47.67	-122.394	1360	4850
17	9.3E+09	20150124	650000	4	3	2950	5000	2	0	3	3	9	1980	970	1979	0	98126	47.5714	-122.375	2140	4000
18	1.88E+09	20140731	395000	3	2	1890	14040	2	0	0	3	7	1890	0	1994	0	98019	47.7277	-121.962	1890	14018
19	6.87E+09	20140529	485000	4	1	1600	4300	1.5	0	0	4	7	1600	0	1916	0	98103	47.6648	-122.343	1610	4300
20	16000397	20141205	189000	2	1	1200	9850	1	0	0	4	7	1200	0	1921	0	98002	47.3089	-122.21	1060	5095
21	7.98E+09	20150424	230000	3	1	1250	9774	1	0	0	4	7	1250	0	1969	0	98003	47.3343	-122.306	1280	8850
22	6.3E+09	20140514	385000	4	1.75	1620	4980	1	0	0	4	7	860	760	1947	0	98133	47.7025	-122.341	1400	4980
23	2.52E+09	20140826	2.00E+06	3	2.75	3050	44867	1	0	4	3	9	2330	720	1968	0	98040	47.5316	-122.233	4110	20336
24	7.14E+09	20140703	285000	5	2.5	2270	6300	2	0	0	3	8	2270	0	1995	0	98092	47.3266	-122.169	2240	7005
25	8.09E+09	20140516	252700	2	1.5	1070	9643	1	0	0	3	7	1070	0	1985	0	98030	47.3533	-122.166	1220	8386
26	3.81E+09	20141120	329000	3	2.25	2450	6500	2	0	0	4	8	2450	0	1985	0	98030	47.3739	-122.172	2200	6865
27	1.2E+09	20141103	233000	3	2	1710	4697	1.5	0	0	5	6	1710	0	1941	0	98002	47.3048	-122.218	1030	4705
28	1.79E+09	20140626	937000	3	1.75	2450	2691	2	0	0	3	8	1750	700	1915	0	98119	47.6386	-122.36	1760	3573
29	3.3E+09	20141201	667000	3	1	1400	1581	1.5	0	0	5	8	1400	0	1909	0	98112	47.6221	-122.314	1860	3861

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
30	5.1E+09	20140624	438000	3	1.75	1520	6380	1	0	0	3	7	790	730	1948	0	98115	47.695	-122.304	1520	6235
31	1.87E+09	20150302	719000	4	2.5	2570	7173	2	0	0	3	8	2570	0	2005	0	98052	47.7073	-122.11	2630	6026
32	8.56E+09	20141110	580500	3	2.5	2320	3980	2	0	0	3	8	2320	0	2003	0	98027	47.5391	-122.07	2580	3980
33	2.43E+09	20141201	280000	2	1.5	1190	1265	3	0	0	3	7	1190	0	2005	0	98133	47.7274	-122.357	1390	1756
34	4.61E+08	20140624	687500	4	1.75	2330	5000	1.5	0	0	4	7	1510	820	1929	0	98117	47.6823	-122.368	1460	5000
35	7.59E+09	20141110	535000	3	1	1090	3000	1.5	0	0	4	8	1090	0	1929	0	98117	47.6889	-122.375	1570	5080
36	7.96E+09	20141203	322500	4	2.75	2060	6659	1	0	0	3	7	1280	780	1981	0	98058	47.4276	-122.157	2020	8720
37	9.55E+09	20140613	696000	3	2.5	2300	3060	1.5	0	0	3	8	1510	790	1930	2002	98115	47.6827	-122.31	1590	3264
38	9.44E+09	20140528	550000	4	1	1660	34848	1	0	0	1	5	930	730	1933	0	98052	47.6621	-122.132	2160	11467
39	2.77E+09	20141230	640000	4	2	2360	6000	2	0	0	4	8	2360	0	1904	0	98107	47.6702	-122.362	1730	4700
40	7.9E+09	20150213	240000	4	1	1220	8075	1	0	0	2	7	890	330	1969	0	98001	47.3341	-122.282	1290	7800
41	2.08E+09	20140620	605000	4	2.5	2620	7553	2	0	0	3	8	2620	0	1996	0	98056	47.5301	-122.18	2620	11884
42	5.55E+09	20140715	625000	4	2.5	2570	5520	2	0	0	3	9	2570	0	2000	0	98074	47.6145	-122.027	2470	5669
43	7.77E+09	20140811	775000	4	2.25	4220	24186	1	0	0	3	8	2600	1620	1984	0	98166	47.445	-122.347	2410	30617
44	7.2E+09	20140707	861990	5	2.75	3595	5639	2	0	0	3	9	3595	0	2014	0	98053	47.6848	-122.016	3625	5639
45	9.27E+09	20141028	685000	3	1	1570	2280	2	0	0	3	7	1570	0	1922	0	98119	47.6413	-122.364	1580	2640
46	1.43E+09	20140729	309000	3	1	1280	9656	1	0	0	4	6	920	360	1959	0	98058	47.4485	-122.175	1340	8808
47	8.04E+09	20140718	488000	3	2.5	3160	13603	2	0	0	3	8	3160	0	2003	0	98019	47.7443	-121.977	3050	9232
48	8.95E+09	20150325	210490	3	1	990	8528	1	0	0	3	6	990	0	1966	0	98023	47.3066	-122.371	1228	8840
49	4.18E+09	20140716	785000	4	2.5	2290	13416	2	0	0	4	9	2290	0	1981	0	98007	47.6194	-122.151	2680	13685
50	9.22E+09	20150428	450000	3	1.75	1250	5963	1	0	0	4	7	1250	0	1953	0	98115	47.6796	-122.301	970	5100
51	8.22E+08	20150311	1.35E+06	3	2.5	2753	65005	1	1	2	5	9	2165	588	1953	0	98070	47.4041	-122.451	2680	72513
52	5.25E+09	20140916	228000	3	1	1190	9199	1	0	0	3	7	1190	0	1955	0	98148	47.4258	-122.322	1190	9364
53	7.23E+09	20150217	345000	5	2.5	3150	9134	1	0	0	4	8	1640	1510	1966	0	98056	47.4934	-122.189	1990	9133
54	7.52E+09	20141231	600000	3	1.75	1410	4080	1	0	0	4	7	1000	410	1950	0	98117	47.6808	-122.384	1410	4080
55	3.63E+09	20150205	585000	2	1.75	1980	8550	1	0	0	3	7	990	990	1981	0	98117	47.6989	-122.369	1480	6738
56	4.22E+09	20150303	920000	5	2.25	2730	6000	1.5	0	0	3	8	2130	600	1927	0	98105	47.6571	-122.281	2730	6000
57	9.82E+09	20140512	885000	4	2.5	2830	5000	2	0	0	3	9	2830	0	1995	0	98105	47.6597	-122.29	1950	5000
58	9.48E+09	20140819	292500	4	2.5	2250	4495	2	0	0	3	7	2250	0	2008	0	98042	47.3663	-122.114	2250	4500

## Appendix B

## The full progress of development, code from jupyter notebook

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
In [2]: df = pd.read_csv('../data/kc_house_data.csv')
```

## Exploratory Data Analysis

```
In [3]: df.isnull().sum()
```

```
Out[3]: id          0
date            0
price           0
bedrooms       0
bathrooms      0
sqft_living    0
sqft_lot       0
floors         0
waterfront     0
view           0
condition      0
grade          0
sqft_above     0
sqft_basement  0
yr_built       0
yr_renovated   0
zipcode        0
lat            0
long           0
sqft_living15  0
sqft_lot15     0
dtype: int64
```

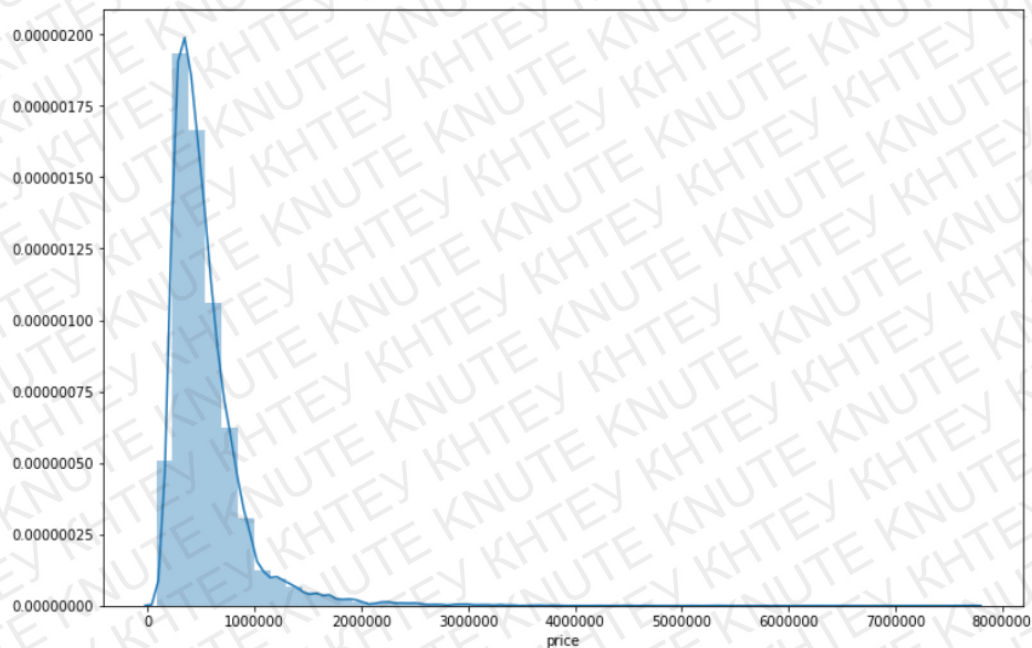
```
In [4]: df.describe().transpose()
```

```
Out[4]:
```

	count	mean	std	min	25%	50%	75%	max
id	21597.0	4.580474e+09	2.876736e+09	1.000102e+06	2.123049e+09	3.904930e+09	7.308900e+09	9.900000e+09
price	21597.0	5.402966e+05	3.673681e+05	7.800000e+04	3.220000e+05	4.500000e+05	6.450000e+05	7.700000e+06
bedrooms	21597.0	3.373200e+00	9.262989e-01	1.000000e+00	3.000000e+00	3.000000e+00	4.000000e+00	3.300000e+01
bathrooms	21597.0	2.115826e+00	7.689843e-01	5.000000e-01	1.750000e+00	2.250000e+00	2.500000e+00	8.000000e+00
sqft_living	21597.0	2.080322e+03	9.181061e+02	3.700000e+02	1.430000e+03	1.910000e+03	2.550000e+03	1.354000e+04
sqft_lot	21597.0	1.509941e+04	4.141264e+04	5.200000e+02	5.040000e+03	7.618000e+03	1.068500e+04	1.651359e+06
floors	21597.0	1.494096e+00	5.396828e-01	1.000000e+00	1.000000e+00	1.500000e+00	2.000000e+00	3.500000e+00
waterfront	21597.0	7.547345e-03	8.654900e-02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	1.000000e+00
view	21597.0	2.342918e-01	7.663898e-01	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	4.000000e+00
condition	21597.0	3.409825e+00	6.505456e-01	1.000000e+00	3.000000e+00	3.000000e+00	4.000000e+00	5.000000e+00
grade	21597.0	7.657915e+00	1.173200e+00	3.000000e+00	7.000000e+00	7.000000e+00	8.000000e+00	1.300000e+01
sqft_above	21597.0	1.788597e+03	8.277598e+02	3.700000e+02	1.190000e+03	1.560000e+03	2.210000e+03	9.410000e+03
sqft_basement	21597.0	2.917250e+02	4.426678e+02	0.000000e+00	0.000000e+00	0.000000e+00	5.600000e+02	4.820000e+03
yr_built	21597.0	1.971000e+03	2.937523e+01	1.900000e+03	1.951000e+03	1.975000e+03	1.997000e+03	2.015000e+03
yr_renovated	21597.0	8.446479e+01	4.018214e+02	0.000000e+00	0.000000e+00	0.000000e+00	0.000000e+00	2.015000e+03
zipcode	21597.0	9.807795e+04	5.351307e+01	9.800100e+04	9.803300e+04	9.806500e+04	9.811800e+04	9.819900e+04
lat	21597.0	4.756009e+01	1.385518e-01	4.715590e+01	4.747110e+01	4.757180e+01	4.767800e+01	4.777760e+01
long	21597.0	-1.222140e+02	1.407235e-01	-1.225190e+02	-1.223280e+02	-1.222310e+02	-1.221250e+02	-1.213150e+02
sqft_living15	21597.0	1.986620e+03	6.852305e+02	3.990000e+02	1.490000e+03	1.840000e+03	2.360000e+03	6.210000e+03
sqft_lot15	21597.0	1.275828e+04	2.727444e+04	6.510000e+02	5.100000e+03	7.620000e+03	1.008300e+04	8.712000e+05

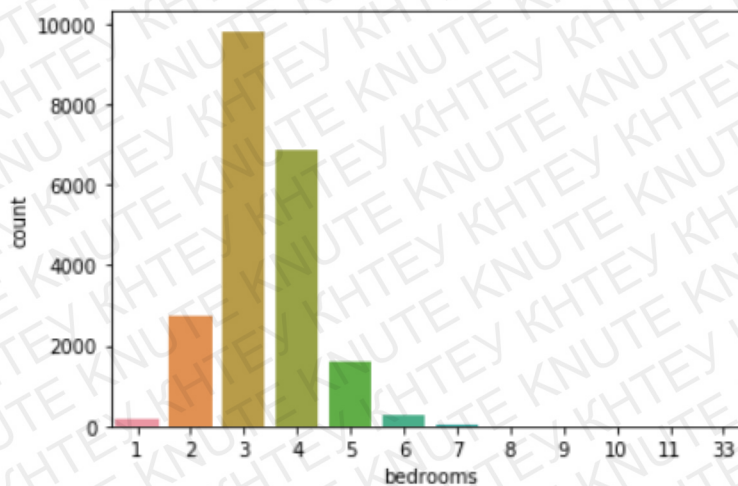
```
In [5]: plt.figure(figsize=(12,8))  
sns.distplot(df['price'])
```

```
Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x26e56791108>
```



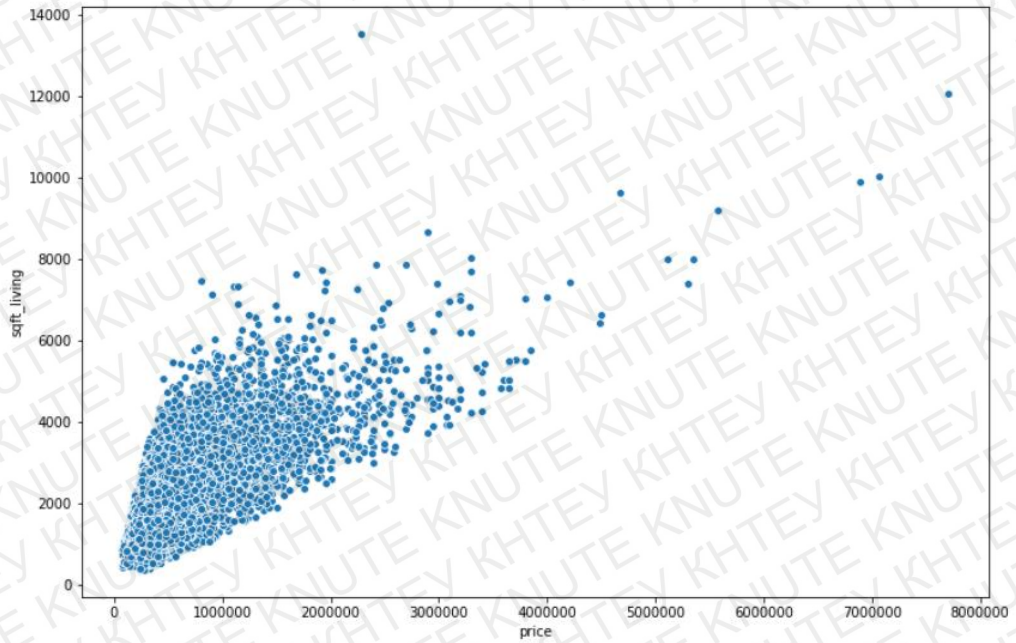
```
In [6]: sns.countplot(df['bedrooms'])
```

```
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x26e56900ec8>
```



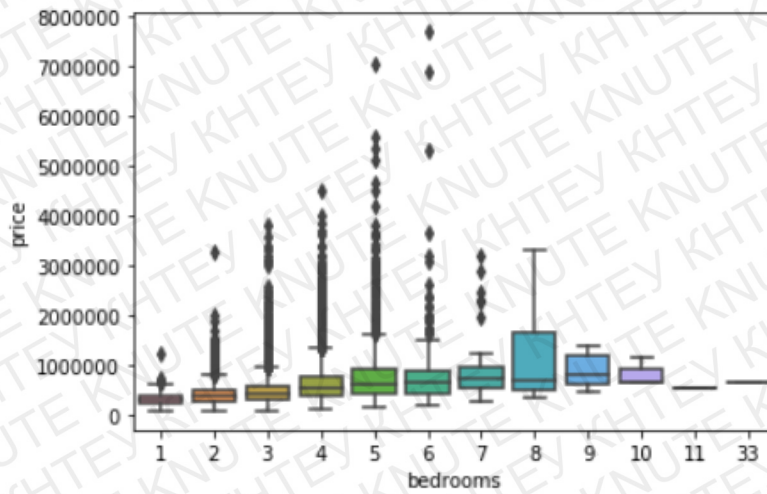
```
In [7]: plt.figure(figsize=(12,8))
sns.scatterplot(x='price',y='sqft_living',data=df)
```

```
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x26e548ed688>
```



```
In [8]: sns.boxplot(x='bedrooms',y='price',data=df)
```

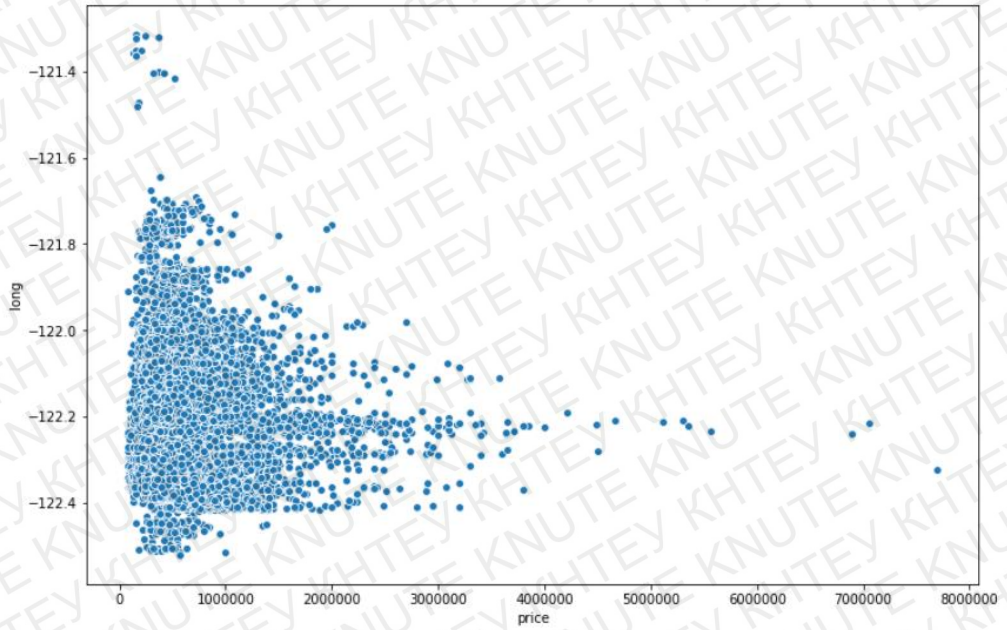
```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x26e569ae388>
```



## Geographical Properties

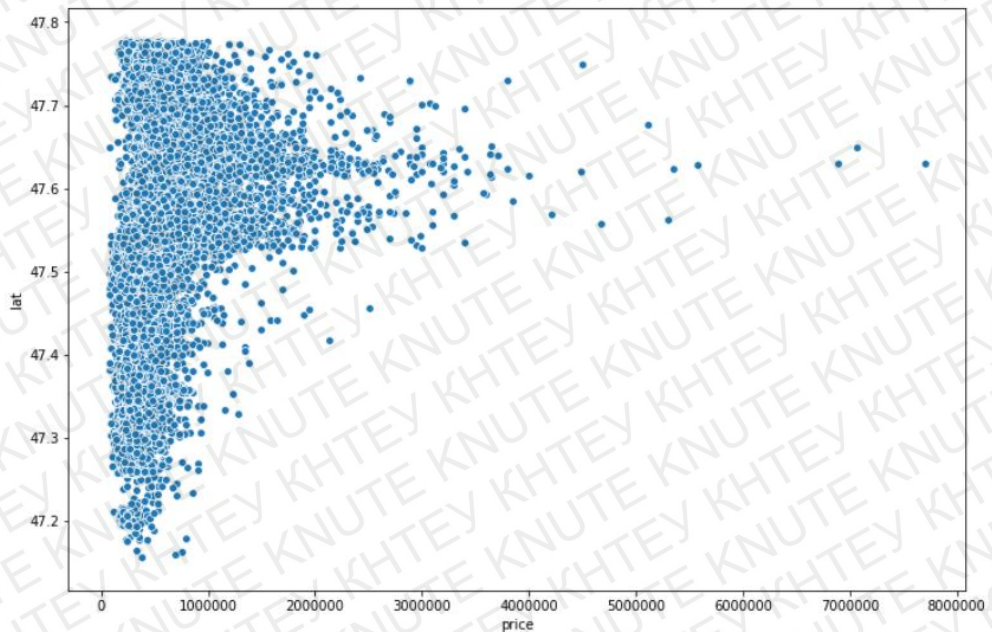
```
In [9]: plt.figure(figsize=(12,8))  
sns.scatterplot(x='price',y='long',data=df)
```

```
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x26e570d1748>
```



```
In [10]: plt.figure(figsize=(12,8))  
sns.scatterplot(x='price',y='lat',data=df)
```

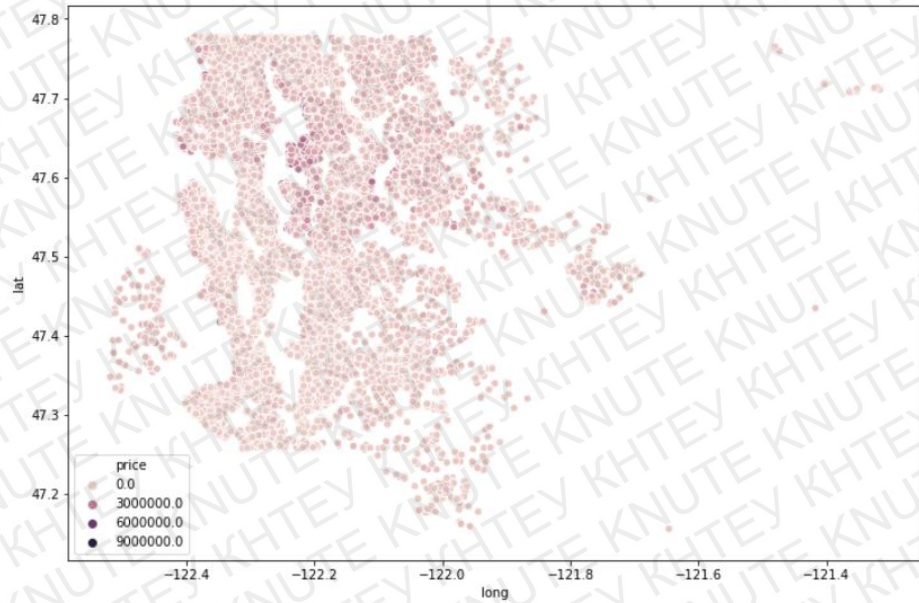
```
Out[10]: <matplotlib.axes._subplots.AxesSubplot at 0x26e57184bc8>
```





```
In [11]: plt.figure(figsize=(12,8))
sns.scatterplot(x='long',y='lat',data=df,hue='price')
```

```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x26e571e78c8>
```



```
In [12]: df.sort_values('price',ascending=False).head(20)
```

```
Out[12]:
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement
	7245	6762700020	10/13/2014	7700000.0	6	8.00	12050	27600	2.5	0	3 ...	13	8570	3480
	3910	9808700762	6/11/2014	7060000.0	5	4.50	10040	37325	2.0	1	2 ...	11	7680	2360
	9245	9208900037	9/19/2014	6890000.0	6	7.75	9890	31374	2.0	0	4 ...	13	8860	1030
	4407	2470100110	8/4/2014	5570000.0	5	5.75	9200	35069	2.0	0	0 ...	13	6200	3000
	1446	8907500070	4/13/2015	5350000.0	5	5.00	8000	23985	2.0	0	4 ...	12	6720	1280
	1313	7558700030	4/13/2015	5300000.0	6	6.00	7390	24829	2.0	1	4 ...	12	5000	2390
	1162	1247600105	10/20/2014	5110000.0	5	5.25	8010	45517	2.0	1	4 ...	12	5990	2020
	8085	1924059029	6/17/2014	4670000.0	5	6.75	9640	13068	1.0	1	4 ...	12	4820	4820
	2624	7738500731	8/15/2014	4500000.0	5	5.50	6640	40014	2.0	1	4 ...	12	6350	290
	8629	3835500195	6/18/2014	4490000.0	4	3.00	6430	27517	2.0	0	0 ...	12	6430	0
	12358	6065300370	5/6/2015	4210000.0	5	6.00	7440	21540	2.0	0	0 ...	12	5550	1890
	4145	6447300265	10/14/2014	4000000.0	4	5.50	7080	16573	2.0	0	0 ...	12	5760	1320
	2083	8106100105	11/14/2014	3850000.0	4	4.25	5770	21300	2.0	1	4 ...	11	5770	0
	7028	853200010	7/1/2014	3800000.0	5	5.50	7050	42840	1.0	0	2 ...	13	4320	2730
	19002	2303900100	9/11/2014	3800000.0	3	4.25	5510	35000	2.0	0	4 ...	13	4910	600
	16288	7397300170	5/30/2014	3710000.0	4	3.50	5550	28078	2.0	0	2 ...	12	3350	2200
	18467	4389201095	5/11/2015	3650000.0	5	3.75	5020	8694	2.0	0	1 ...	12	3970	1050
	6502	4217402115	4/21/2015	3650000.0	6	4.75	5480	19401	1.5	1	4 ...	11	3910	1570
	15241	2425049063	9/11/2014	3640000.0	4	3.25	4830	22257	2.0	1	4 ...	11	4830	0
	19133	3625049042	10/11/2014	3640000.0	5	6.00	5490	19897	2.0	0	0 ...	12	5490	0

20 rows × 21 columns

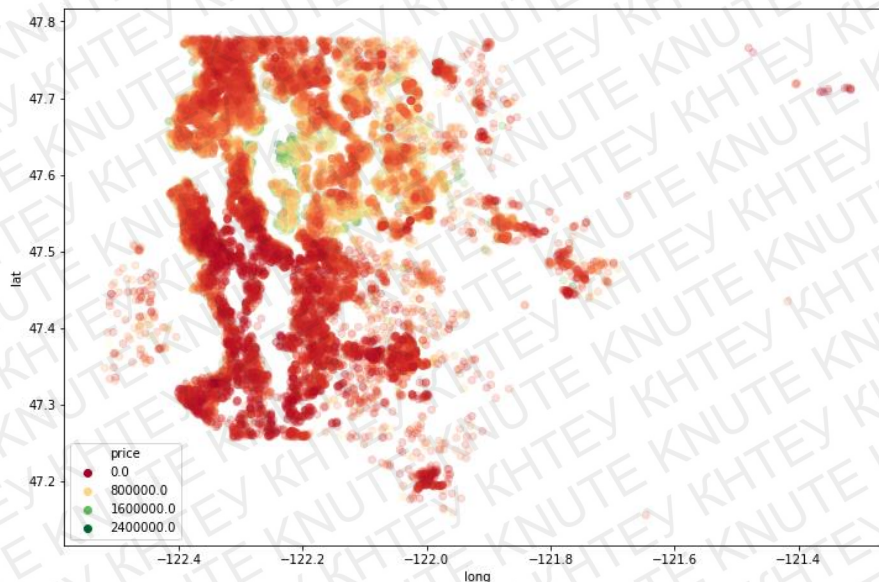
```
In [13]: len(df)*(0.01)
```

```
Out[13]: 215.97
```

```
In [14]: non_top_1_perc = df.sort_values('price',ascending=False).iloc[216:]
```

```
In [15]: plt.figure(figsize=(12,8))
sns.scatterplot(x='long',y='lat',
               data=non_top_1_perc,hue='price',
               palette='RdYlGn',edgecolor=None,alpha=0.2)
```

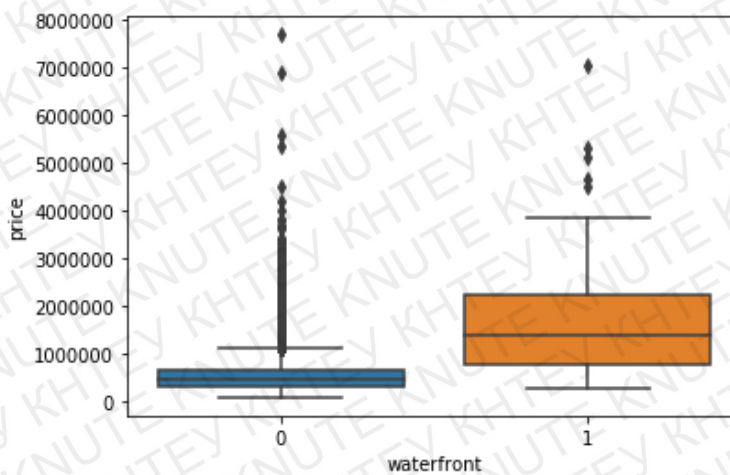
```
Out[15]: <matplotlib.axes._subplots.AxesSubplot at 0x26e578a5388>
```



## Other Features

```
In [16]: sns.boxplot(x='waterfront',y='price',data=df)
```

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x26e5794c4c8>
```



## Working with Feature Data

In [17]: `df.head()`

```
Out[17]:
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...	grade	sqft_above	sqft_basement	yr_built	yr_re
0	7129300520	10/13/2014	221900.0	3	1.00	1180	5650	1.0	0	0	...	7	1180	0	1955	
1	6414100192	12/9/2014	538000.0	3	2.25	2570	7242	2.0	0	0	...	7	2170	400	1951	
2	5631500400	2/25/2015	180000.0	2	1.00	770	10000	1.0	0	0	...	6	770	0	1933	
3	2487200875	12/9/2014	604000.0	4	3.00	1960	5000	1.0	0	0	...	7	1050	910	1965	
4	1954400510	2/18/2015	510000.0	3	2.00	1680	8080	1.0	0	0	...	8	1680	0	1987	

5 rows × 21 columns

In [18]: `df.info()`

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21597 entries, 0 to 21596
Data columns (total 21 columns):
id                21597 non-null int64
date             21597 non-null object
price            21597 non-null float64
bedrooms         21597 non-null int64
bathrooms        21597 non-null float64
sqft_living      21597 non-null int64
sqft_lot         21597 non-null int64
floors           21597 non-null float64
waterfront       21597 non-null int64
view             21597 non-null int64
condition        21597 non-null int64
grade            21597 non-null int64
sqft_above       21597 non-null int64
sqft_basement    21597 non-null int64
yr_built         21597 non-null int64
yr_renovated     21597 non-null int64
zipcode          21597 non-null int64
lat              21597 non-null float64
long             21597 non-null float64
sqft_living15    21597 non-null int64
sqft_lot15       21597 non-null int64
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
```

In [19]: `df = df.drop('id', axis=1)`

In [20]: `df.head()`

```
Out[20]:
```

	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovate
0	10/13/2014	221900.0	3	1.00	1180	5650	1.0	0	0	3	7	1180	0	1955	
1	12/9/2014	538000.0	3	2.25	2570	7242	2.0	0	0	3	7	2170	400	1951	199
2	2/25/2015	180000.0	2	1.00	770	10000	1.0	0	0	3	6	770	0	1933	
3	12/9/2014	604000.0	4	3.00	1960	5000	1.0	0	0	5	7	1050	910	1965	
4	2/18/2015	510000.0	3	2.00	1680	8080	1.0	0	0	3	8	1680	0	1987	

## Feature Engineering from Date

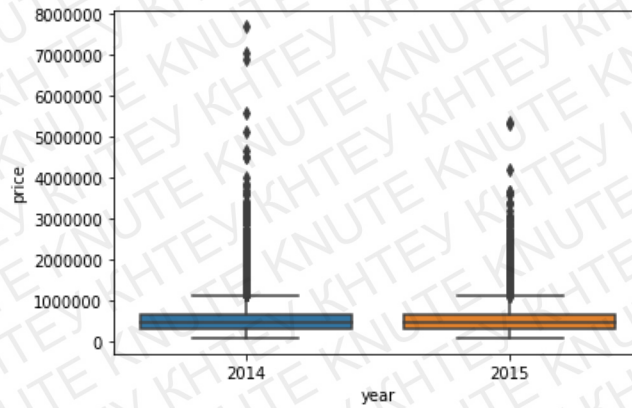
```
In [21]: df['date'] = pd.to_datetime(df['date'])
```

```
In [22]: df['month'] = df['date'].apply(lambda date:date.month)
```

```
In [23]: df['year'] = df['date'].apply(lambda date:date.year)
```

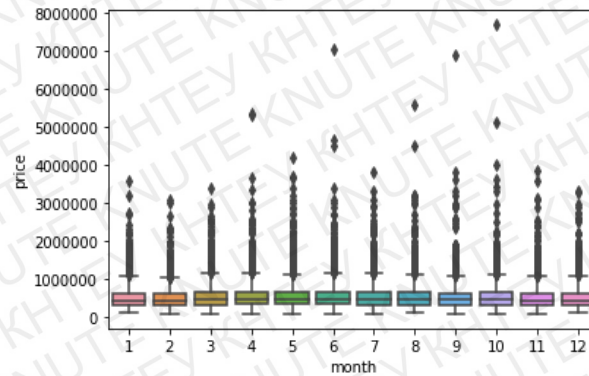
```
In [24]: sns.boxplot(x='year',y='price',data=df)
```

```
Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x26e59114848>
```



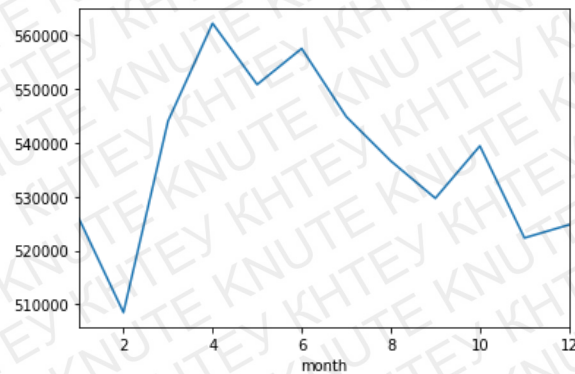
```
In [25]: sns.boxplot(x='month',y='price',data=df)
```

```
Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x26e593b1f48>
```



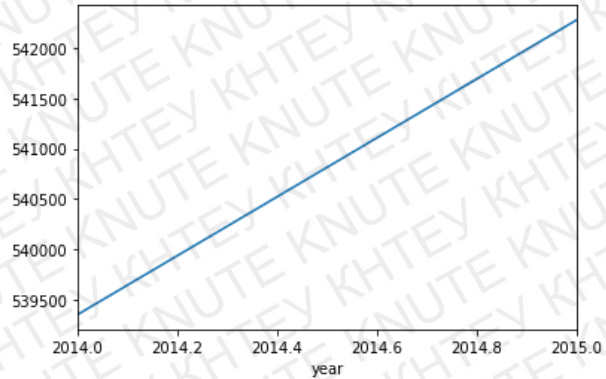
```
In [26]: df.groupby('month').mean()['price'].plot()
```

```
Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x26e593bbc48>
```



```
In [27]: df.groupby('year').mean()['price'].plot()
```

```
Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x26e5959ed48>
```



```
In [28]: df = df.drop('date', axis=1)
```

```
In [29]: df.columns
```

```
Out[29]: Index(['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', 'month', 'year'],
             dtype='object')
```

```
In [30]: df['zipcode'].value_counts()
```

```
Out[30]: 98103    602
          98038    589
          98115    583
          98052    574
          98117    553
          ...
          98102    104
          98010    100
          98024     80
          98148     57
          98039     50
          Name: zipcode, Length: 70, dtype: int64
```

```
In [31]: df = df.drop('zipcode', axis=1)
```

```
In [32]: df.head()
```

```
Out[32]:
```

	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition	grade	sqft_above	sqft_basement	yr_built	yr_renovated	lat
0	221900.0	3	1.00	1180	5650	1.0	0	0	3	7	1180	0	1955	0	47.5112
1	538000.0	3	2.25	2570	7242	2.0	0	0	3	7	2170	400	1951	1991	47.7210
2	180000.0	2	1.00	770	10000	1.0	0	0	3	6	770	0	1933	0	47.7379
3	604000.0	4	3.00	1960	5000	1.0	0	0	5	7	1050	910	1965	0	47.5208
4	510000.0	3	2.00	1680	8080	1.0	0	0	3	8	1680	0	1987	0	47.6168

```
In [33]: df['yr_renovated'].value_counts()
```

```
Out[33]: 0      20683
          2014     91
          2013     37
          2003     36
          2000     35
          ...
          1934      1
          1959      1
          1951      1
          1948      1
          1944      1
          Name: yr_renovated, Length: 70, dtype: int64
```

```
In [34]: df['sqft_basement'].value_counts()
```

```
Out[34]: 0      13110
         600      221
         700      218
         500      214
         800      206
         ...
         792        1
         2590       1
         935        1
         2390       1
         248        1
         Name: sqft_basement, Length: 306, dtype: int64
```

### Scaling

```
In [38]: from sklearn.preprocessing import MinMaxScaler
```

```
In [39]: scaler = MinMaxScaler()
```

```
In [40]: X_train = scaler.fit_transform(X_train)
```

```
In [41]: X_test = scaler.transform(X_test)
```

```
In [42]: X_train.shape
```

```
Out[42]: (15117, 19)
```

```
In [43]: X_test.shape
```

```
Out[43]: (6480, 19)
```

### Creating a Model

```
In [44]: from tensorflow.keras.models import Sequential
         from tensorflow.keras.layers import Dense, Activation
         from tensorflow.keras.optimizers import Adam
```

```
In [45]: model = Sequential()
```

```
model.add(Dense(19, activation='relu'))
model.add(Dense(19, activation='relu'))
model.add(Dense(19, activation='relu'))
model.add(Dense(19, activation='relu'))
model.add(Dense(1))
```

```
model.compile(optimizer='adam', loss='mse')
```

### Training the Model

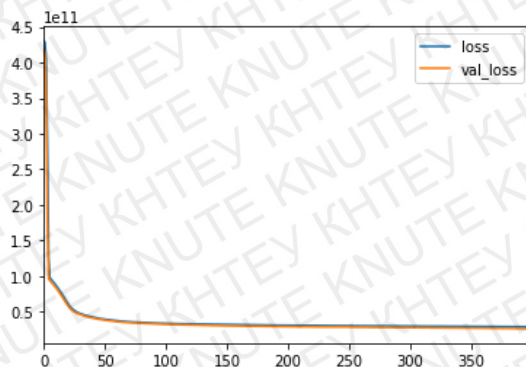
```
In [46]: model.fit(x=X_train, y=y_train.values,
                  validation_data=(X_test, y_test.values),
                  batch_size=128, epochs=400)
```

```
Train on 15117 samples, validate on 6480 samples
Epoch 1/400
15117/15117 [=====] - 1s 74us/sample - loss: 430228839929.3955 - val_loss: 418844493758.2617
Epoch 2/400
15117/15117 [=====] - 0s 21us/sample - loss: 428253102583.9730 - val_loss: 411996069786.8642
Epoch 3/400
15117/15117 [=====] - 0s 21us/sample - loss: 401401261499.3472 - val_loss: 351554389222.0839
Epoch 4/400
15117/15117 [=====] - 0s 22us/sample - loss: 281418027052.1653 - val_loss: 178278463958.2815
Epoch 5/400
15117/15117 [=====] - 0s 21us/sample - loss: 126127754575.5078 - val_loss: 95946680446.4198
Epoch 6/400
15117/15117 [=====] - 0s 21us/sample - loss: 97670204498.9794 - val_loss: 93708013133.1160
Epoch 7/400
15117/15117 [=====] - 0s 21us/sample - loss: 95694764310.6754 - val_loss: 91793937901.0370
Epoch 8/400
15117/15117 [=====] - 0s 22us/sample - loss: 93689741109.3270 - val_loss: 89857905297.3827
Epoch 9/400
15117/15117 [=====] - 0s 22us/sample - loss: 91635817511.3560 - val_loss: 87934673219.6346
```

```
In [47]: losses = pd.DataFrame(model.history.history)
```

```
In [48]: losses.plot()
```

```
Out[48]: <matplotlib.axes._subplots.AxesSubplot at 0x2707c650b88>
```



```
In [49]: from sklearn.metrics import mean_squared_error, mean_absolute_error, explained_variance_score
```

### Predicting on Brand New Data

```
In [50]: X_test
```

```
Out[50]: array([[0.1      , 0.08      , 0.04239917, ..., 0.00887725, 0.63636364,
                0.        ],
                [0.3      , 0.36      , 0.17269907, ..., 0.00993734, 0.81818182,
                0.        ],
                [0.2      , 0.24      , 0.12512927, ..., 0.00547073, 0.90909091,
                0.        ],
                ...,
                [0.1      , 0.08      , 0.05584281, ..., 0.00506255, 1.        ,
                0.        ],
                [0.3      , 0.2      , 0.22233713, ..., 0.00774485, 0.09090909,
                1.        ],
                [0.3      , 0.32      , 0.27611169, ..., 0.0196531 , 0.45454545,
                0.        ]])
```

```
In [51]: predictions = model.predict(X_test)
```

```
In [52]: mean_absolute_error(y_test, predictions)
```

```
Out[52]: 101666.74811137635
```

```
In [53]: np.sqrt(mean_squared_error(y_test, predictions))
```

```
Out[53]: 162549.26352210157
```

```
In [54]: explained_variance_score(y_test, predictions)
```

```
Out[54]: 0.8009472831860807
```

```
In [55]: df['price'].mean()
```

```
Out[55]: 540296.5735055795
```

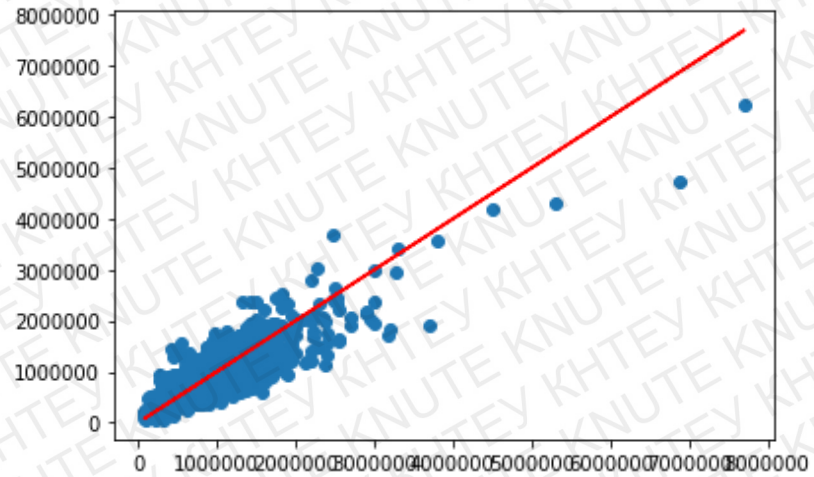
```
In [56]: df['price'].median()
```

```
Out[56]: 450000.0
```

```
In [57]: # Our predictions
plt.scatter(y_test, predictions)

# Perfect predictions
plt.plot(y_test, y_test, 'r')
```

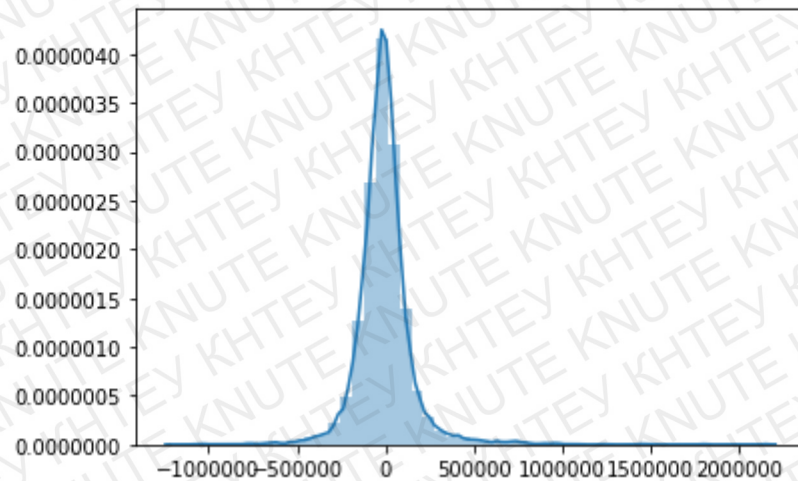
Out[57]: [[matplotlib.lines.Line2D](#) at 0x2707a07fe08]



```
In [58]: errors = y_test.values.reshape(6480, 1) - predictions
```

```
In [59]: sns.distplot(errors)
```

Out[59]: [matplotlib.axes.\\_subplots.AxesSubplot](#) at 0x270819ff5c8





## Predicting on a brand new house

```
In [81]: single_house = df.drop('price',axis=1).iloc[0]
```

```
In [86]: single_house = scaler.transform(single_house.values.reshape(-1, 19))
```

```
In [87]: single_house
```

```
Out[87]: array([[0.22222222, 0.08      , 0.11392405, 0.00433777, 0.
,
0.      , 0.      , 0.5      , 0.375    , 0.11654676,
0.      , 0.47826087, 0.      , 0.56914119, 0.21760797,
0.16510319, 0.00574235, 0.81818182, 0.      ]])
```

```
In [89]: model.predict(single_house)
```

```
Out[89]: array([[283862.12]], dtype=float32)
```

```
Out[90]: price          271900.0000
bedrooms             3.0000
bathrooms            1.0000
sqft_living          1180.0000
sqft_lot             5650.0000
floors                1.0000
waterfront           0.0000
view                  0.0000
condition            3.0000
grade                7.0000
sqft_above           1180.0000
sqft_basement         0.0000
yr_built             1955.0000
yr_renovated         0.0000
lat                   47.5112
long                 -122.2570
sqft_living15        1340.0000
sqft_lot15           5650.0000
month                 10.0000
year                  2014.0000
Name: 0, dtype: float64
```