
Welcome

Welcome to The Taste of Data Science!

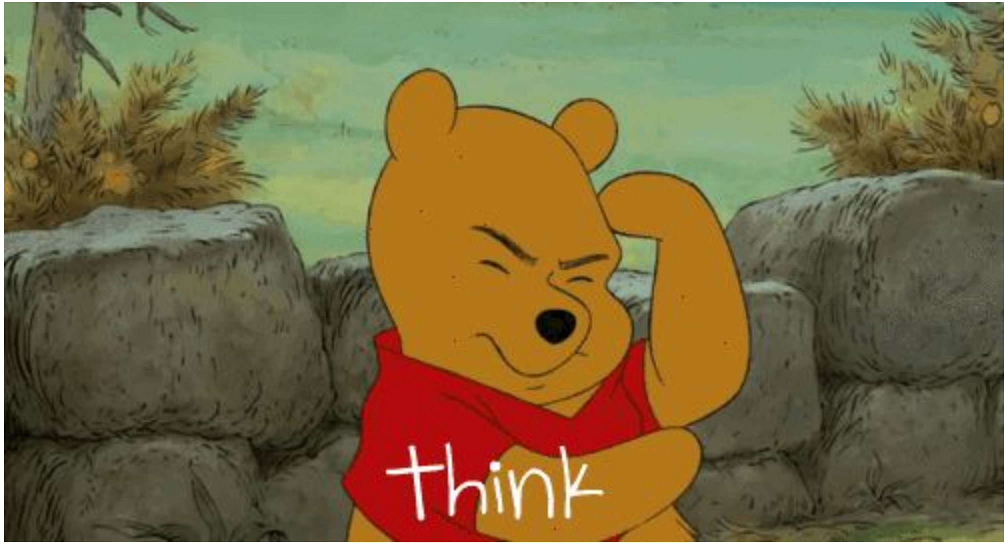
Data Science in Python &
Machine Learning

Learning Goals

By the end of this session, participants will be able to

- Identify what data science is
- Describe what data scientists do & learning goals for aspiring data scientists
- Summarize the types of machine learning and when they are used
- Describe the machine learning model life cycle & walkthrough a business problem

In your own words, what is data science to you?

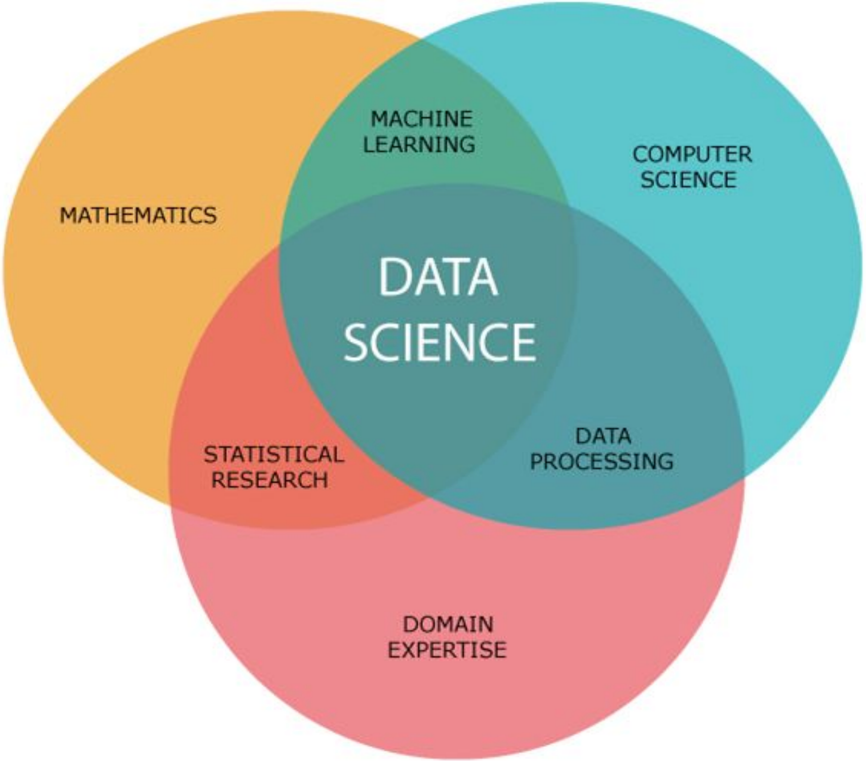


What is Data Science Anyway?

What is Data Science?

"**Data science** is an **interdisciplinary field** that uses **scientific methods, processes,** algorithms and systems to **extract knowledge and insights** from noisy, structured and unstructured data and apply knowledge and **actionable insights** from data across a **broad range of application domains**…."

- [Wikipedia's Definition of Data Science]



[Image thanks to Serap Baysal]

Why Do We Need Data Science?

Data Scientists are key in the decision making process.

They are beneficial for the following tasks:

- Collecting Data
- Cleaning Data
- Visualizing Data
- Modeling Data
- Making Future Predictions

What do you think Data Scientists do?

Check all that apply.

What do Data Scientists Do?

Data Scientists:

- Gather and prepare data for use in analytics applications
- Use various types of analytics tools to detect patterns, trends and relationships in data sets
- Develop statistical and predictive models to provide actionable insights
- Create data visualizations, dashboards and reports to communicate their findings

Image thanks to Serap Baysal

What do you need to learn to become a Data Scientist?

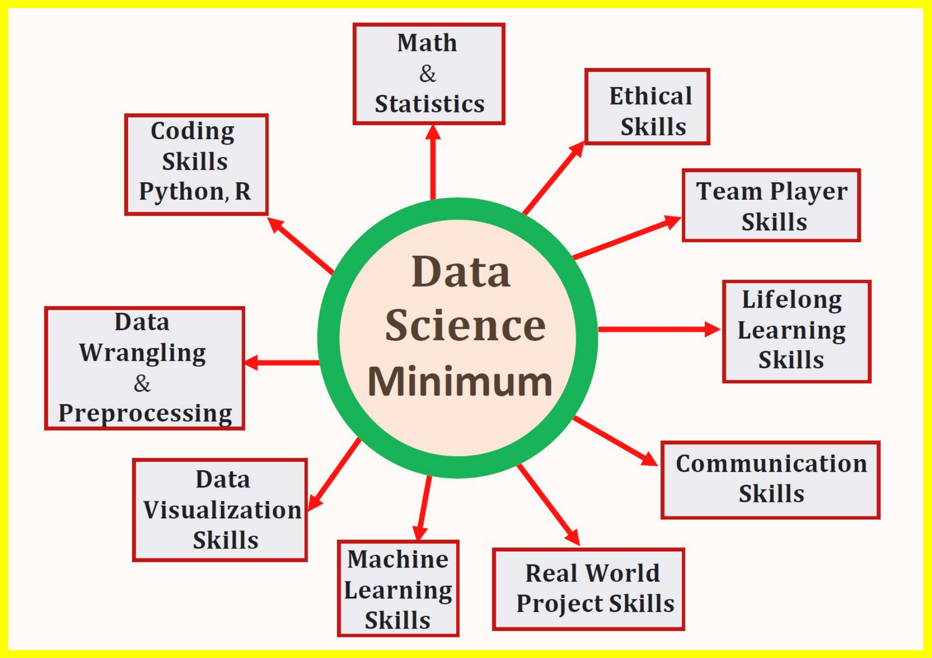


Image thanks to Benjamin Obi Tayo, Ph.D

Coding Dojo Ninja to the Rescue!!!

Coding Dojo DS Program Schedule:

16-Week Program

Weeks 1 - 4
Data Science Fundamentals

Weeks 5 - 8
Machine Learning

Weeks 9 - 12
Advanced Machine Learning

Weeks 13 - 16
Data Enrichment

20-Week Program

Weeks 1 - 4
Data Science Fundamentals

Weeks 5 - 8
Machine Learning

Weeks 9 - 12
Advanced Machine Learning

Weeks 13 - 16
Data Enrichment

Weeks 17-20
Data Visualization

Programs consist of 4-Week Courses

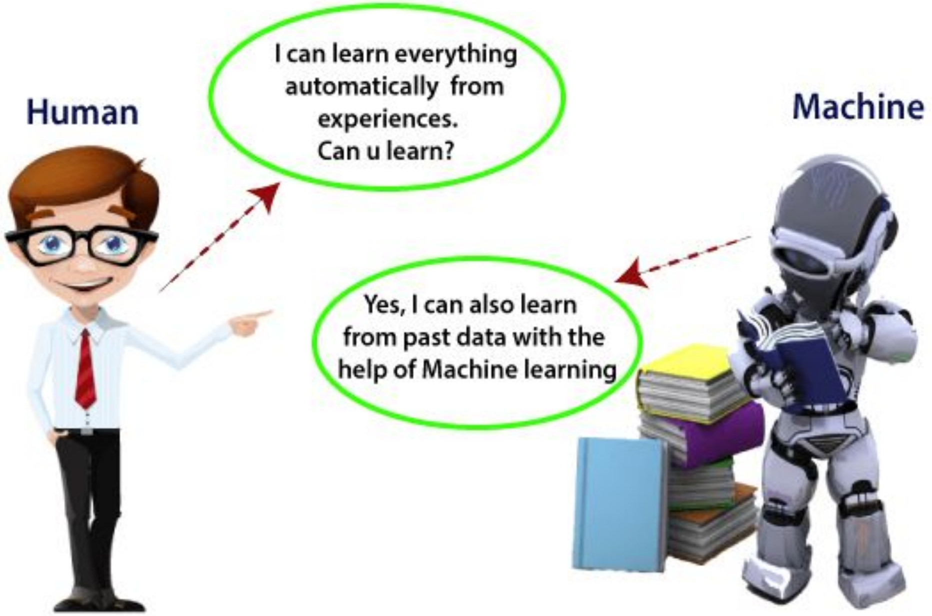
Machine Learning & Applications

Python Packages/Modules for Data Science

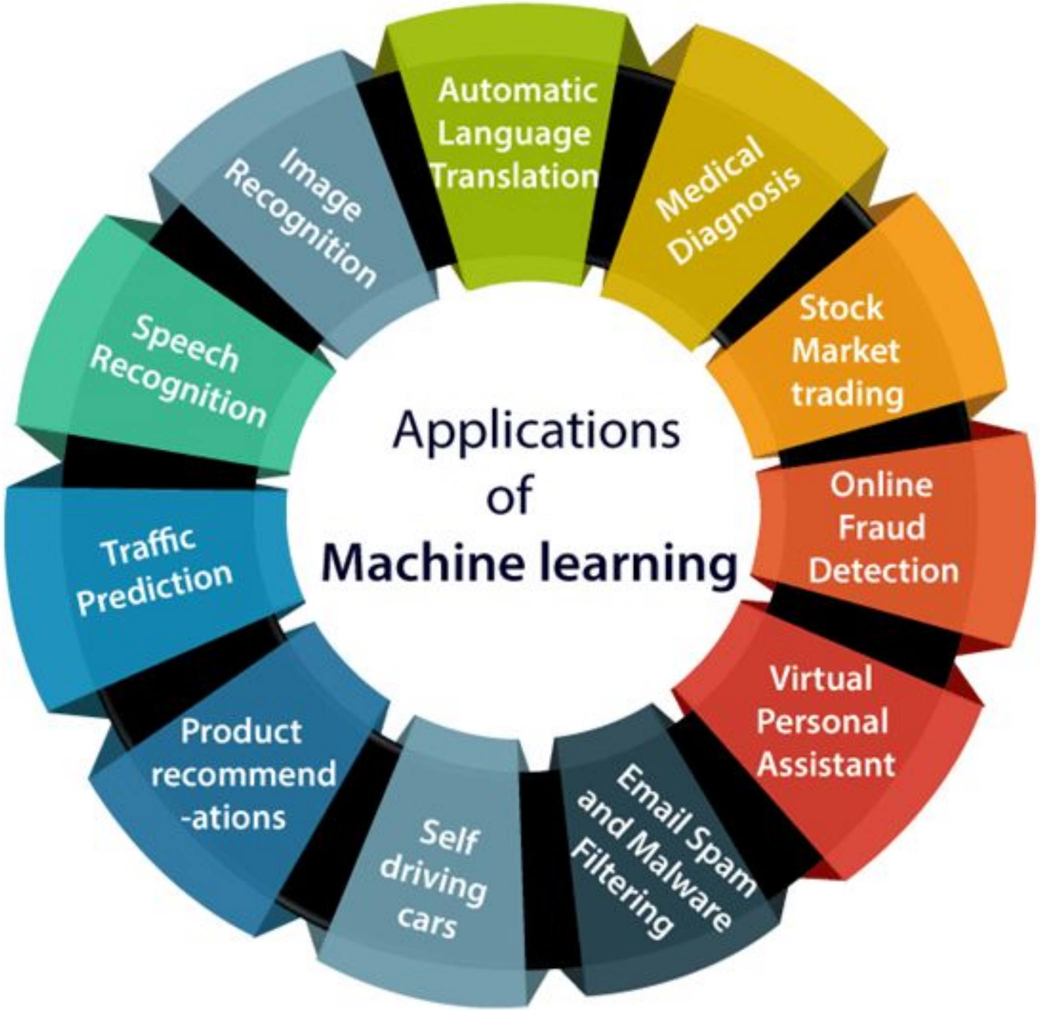
- Python comes with many built-in functions
    - [Built-In Functions]
- Most of the functionality needed as data scientists is not included in base Python.
- We can download other collections of functions and classes, called Packages (A.K.A. Libraries, A.K.A. Modules)
- A few examples of packages used by data scientists:
    - Pandas
    - Numpy
    - Seaborn
    - Matplotlib
    - Sci-kit Learn (Sklearn)

What is Machine Learning?

Machine learning is an application of artificial intelligence that provides systems the ability to automatically learn and improve from experience without being explicitly programmed.



In chat, list at least 1 application of machine learning that you are familiar with.


Applications of Machine learning
Automatic Language Translation
Medical Diagnosis
Stock Market trading
Online Fraud Detection
Virtual Personal Assistant
Email Spam and Malware Filtering
Self driving cars
Product recommend -ations
Traffic Prediction
Speech Recognition
Image Recognition

Types of Machine Learning

Machine Learning

Supervised Learning

Unsupervised Learning

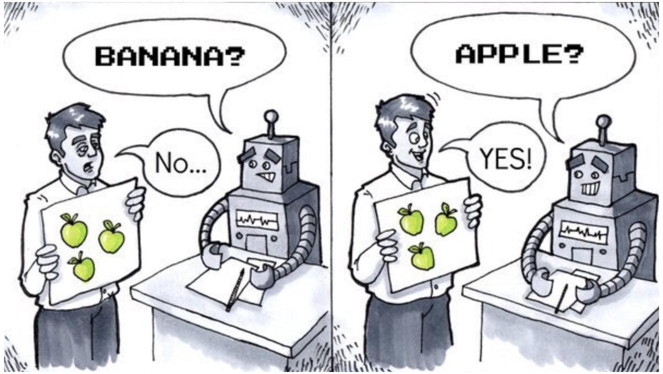
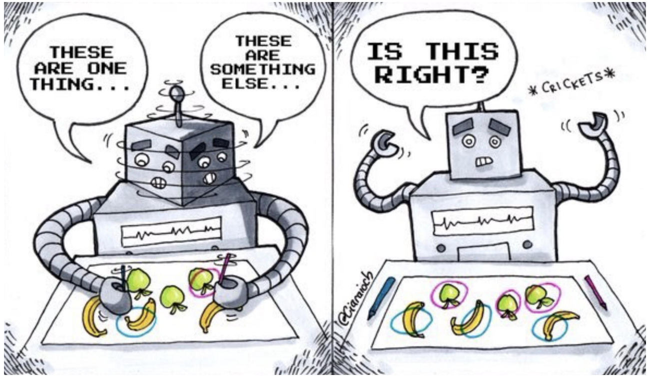
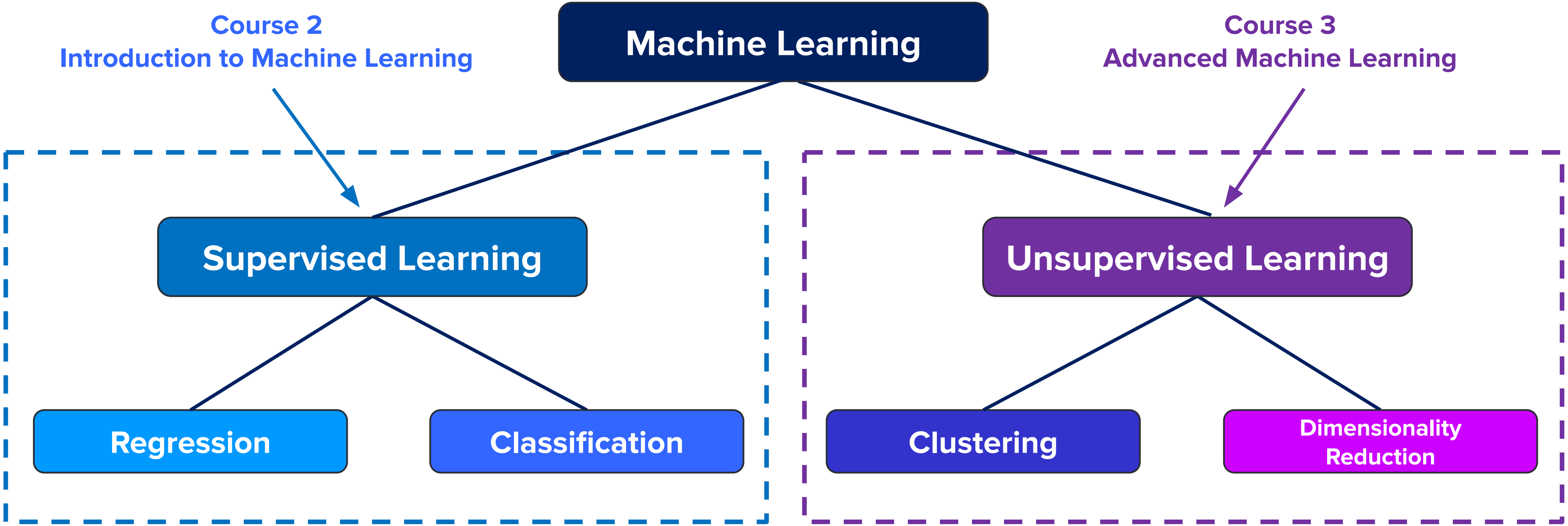


Image-Source



Types of Machine Learning



Supervised Machine Learning
Regression

- **When the goal is to predict a continuous numeric variable, such as a float.**
- **A model prediction can be any real number!**

Supervised Machine Learning
Regression

What examples can you think of?
List them in chat.

Supervised Machine Learning
Regression

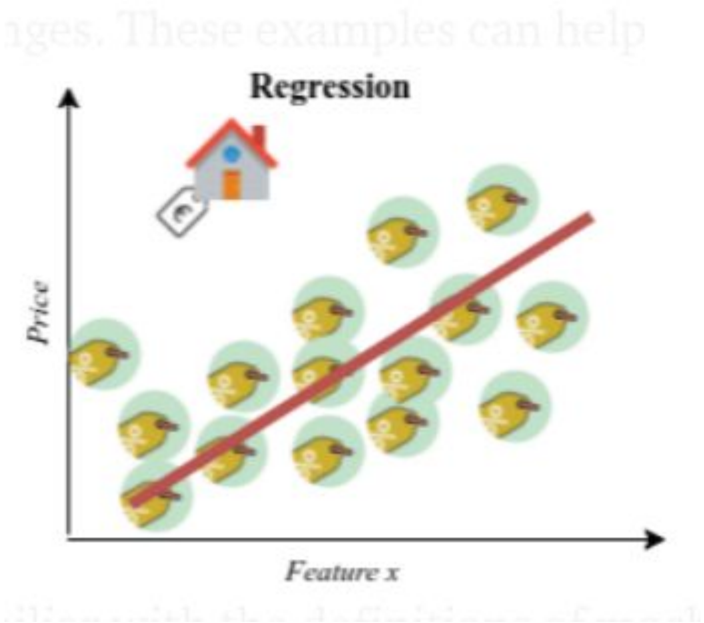
Examples:

- **Home Sale Prices**
- **Total Sales**
- **Stock Price Predictions**
- **Ages**
- **Height / Weight / Length**
- **Temperature**

Supervised Machine Learning
Regression

Number of Square Feet	Selling Price
1000	$100,000
1500	$150,000
2000	$200,000
3000	$300,000

Selling Price

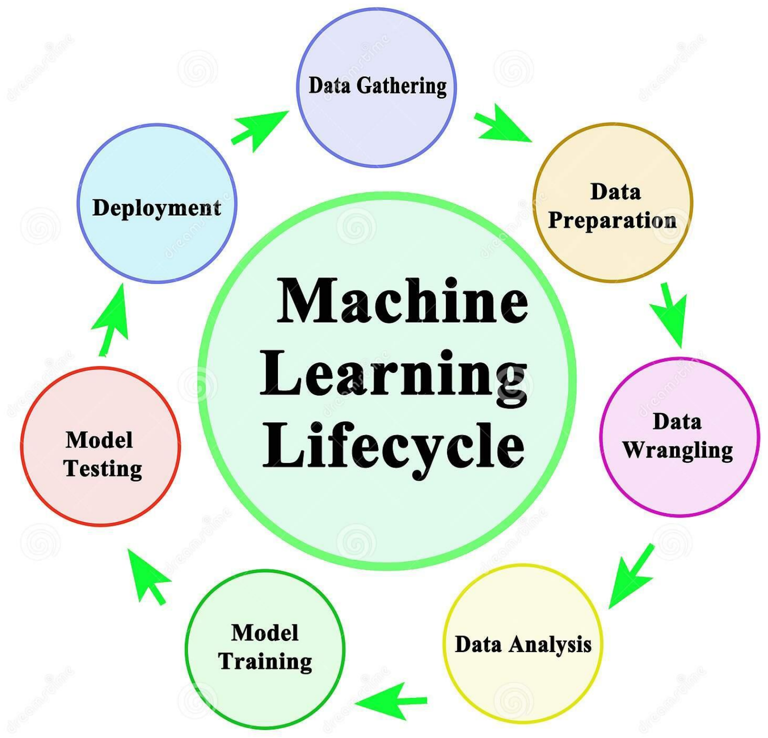


Number of Square Feet

Machine Learning Life Cycle

Machine Learning Life Cycle

Data Gathering	The goal of this steps is to obtain and identify all data related problems. For example, Identify data sources, integration of data from different sources.
Data Preparation	We will put our data at suitable place, then start exploring it, like how many records, quality, types of data etc.
Data Wrangling	This is process to clean and converting data into usable form.
Data Analysis	Here we decide which model , techniques we are going to use.
Model Training	Here we fit and train model to check the performance of the model.
Model Testing	Here we check if our trained model is working good on test data.
Deployment	We deploy the model in real world.



Seattle King County, WA
Housing Market Analysis and Modeling

BUSINESS PROBLEM

You are hired by Seattle based real estate agency **to generate the model which predicts the prices of houses in King County, WA** based on certain property features

Data Gathering

Sources

- Kaggle - King County ,WA Housing Data (May 2014- May 2015)
- GeoDa Data and Lab
- UCI Machine Learning Repository
- Real Estate Agency like Zillow, Redfin, Realtor etc.. also allow you to use their data

Data Dictionary

The dataset contains the following columns(Features):

- **id:** A unique sale id relating to a house sale
- **date:** Date of house sale
- **price:** The price which the house sold for
- **bedrooms:** How many bedrooms the house has
- **bathrooms:** How many bathrooms the house has
- **sqft_living:** How much square footage the house has
- **sqft_lot:** How much square footage the lot has
- **floors:** How many floors the house has
- **waterfront:** Whether the house is on the waterfront. Originally contained 'YES' or 'NO', converted to 0 or 1 for comparative purposes
- **view:** Whether the house has a view and whether it's fair, average, good, or excellent.
- **condition:** overall condition of the house: Poor, Fair, Average, Good, Very Good
- **grade**: Numerical grading for house
- **sqft_above**: How much of the houses square footage is above ground
- **sqft_basement**: How much of the square footage is in the basement
- **yr_built**: Year the house was built
- **yr_renovated**: Year the house was renovated, if applicable
- **zip code:** House zipcode
- lat: House's latitude coordinate
- long: House's longitude coordinate
- **sqft_living15**: Average size of living space for the closest 15 houses
- **sqft_lot15**: Average size of lot for the closest 15 houses

Data dictionary in Python

```
df =pd.read_csv('Data/kc_house_data.csv')
```

```
df.head()
```

	id	date	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	...
0	7129300520	20141013T000000	221,900.0000	3	1.0000	1180	5650	1.0000	0	0	...
1	6414100192	20141209T000000	538,000.0000	3	2.2500	2570	7242	2.0000	0	0	...
2	5631500400	20150225T000000	180,000.0000	2	1.0000	770	10000	1.0000	0	0	...
3	2487200875	20141209T000000	604,000.0000	4	3.0000	1960	5000	1.0000	0	0	...
4	1954400510	20150218T000000	510,000.0000	3	2.0000	1680	8080	1.0000	0	0	...

5 rows × 21 columns

Data Inspection

```
1  df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   id             21613 non-null  int64
 1   date           21613 non-null  object
 2   price          21613 non-null  float64
 3   bedrooms       21613 non-null  int64
 4   bathrooms      21613 non-null  float64
 5   sqft_living    21613 non-null  int64
 6   sqft_lot       21613 non-null  int64
 7   floors         21613 non-null  float64
 8   waterfront     21613 non-null  int64
 9   view           21613 non-null  int64
 10  condition      21613 non-null  int64
 11  grade          21613 non-null  int64
 12  sqft_above     21613 non-null  int64
 13  sqft_basement  21613 non-null  int64
 14  yr_built       21613 non-null  int64
 15  yr_renovated   21613 non-null  int64
 16  zipcode        21613 non-null  int64
 17  lat            21613 non-null  float64
 18  long           21613 non-null  float64
 19  sqft_living15  21613 non-null  int64
 20  sqft_lot15     21613 non-null  int64
dtypes: float64(5), int64(15), object(1)
memory usage: 3.5+ MB
```

- 21 - features related to houses
- 21613 - data points

Data Inspection

```
# Numeric columns
df.describe()
```

	id	price	bedrooms	bathrooms	sqft_living	sqft_lot	floors	waterfront	view	condition
count	21,613.0000	21,613.0000	21,613.0000	21,613.0000	21,613.0000	21,613.0000	21,613.0000	21,613.0000	21,613.0000	21,613.0000
mean	4,580,301,520.8650	540,088.1418	3.3708	2.1148	2,079.8997	15,106.9676	1.4943	0.0075	0.2343	3.4094
std	2,876,565,571.3121	367,127.1965	0.9301	0.7702	918.4409	41,420.5115	0.5400	0.0865	0.7663	0.6507
min	1,000,102.0000	75,000.0000	0.0000	0.0000	290.0000	520.0000	1.0000	0.0000	0.0000	1.0000
25%	2,123,049,194.0000	321,950.0000	3.0000	1.7500	1,427.0000	5,040.0000	1.0000	0.0000	0.0000	3.0000
50%	3,904,930,410.0000	450,000.0000	3.0000	2.2500	1,910.0000	7,618.0000	1.5000	0.0000	0.0000	3.0000
75%	7,308,900,445.0000	645,000.0000	4.0000	2.5000	2,550.0000	10,688.0000	2.0000	0.0000	0.0000	4.0000
max	9,900,000,190.0000	7,700,000.0000	33.0000	8.0000	13,540.0000	1,651,359.0000	3.5000	1.0000	4.0000	5.0000

Data Preparation

Check for any Null values

```
# check for missing/null values
df.isna().sum()
```

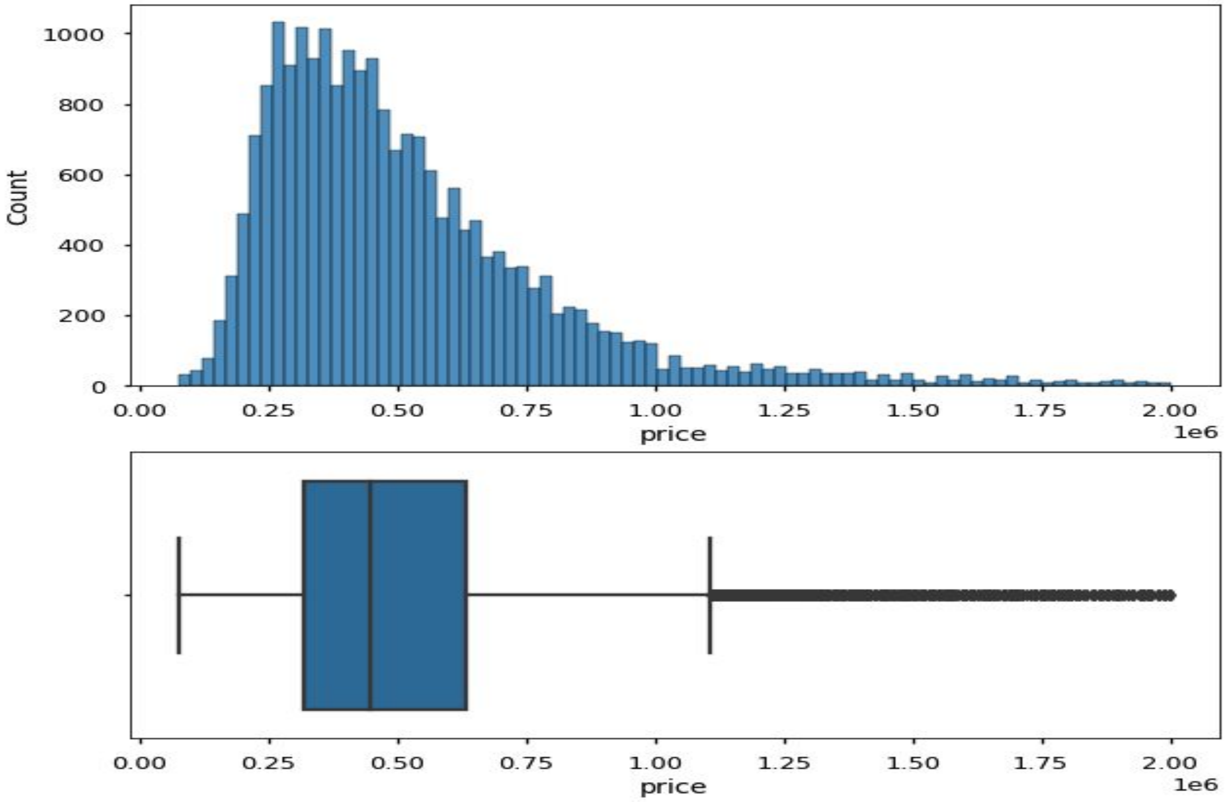
```
date             0
price            0
bedrooms         0
bathrooms        0
sqft_living      0
sqft_lot         0
floors           0
waterfront       0
view             0
condition        0
grade            0
sqft_above       0
sqft_basement    0
yr_built         0
yr_renovated     0
zipcode    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

Check for duplicate values

```
# Check for duplicates
df.duplicated().sum()
```

```
0
```

Check for outliers in our Target (Price)



Data Wrangling

- Transform the raw data to the easily accessible format based on the requirements
  - Ex : Feature Engineering - derived separate month and year from date column

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 21613 entries, 0 to 21612
Data columns (total 21 columns):
 #    Column          Non-Null Count   Dtype
---   ------          --------------   -----
 0    id              21613 non-null   int64
 1    date            21613 non-null   object
```

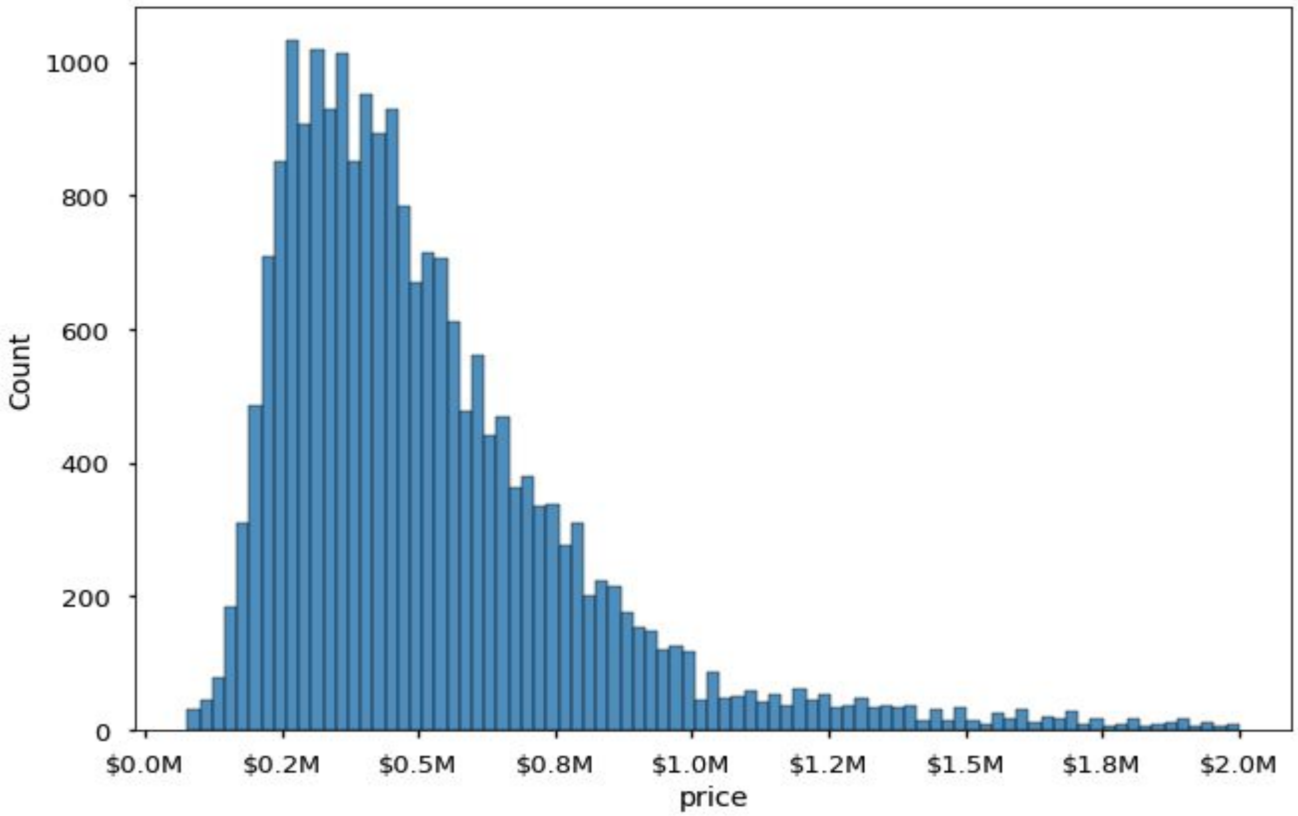
datetime	Month_name	Month	Year
2014-10-13	October	10	2014
2014-12-09	December	12	2014
2015-02-25	February	2	2015
2014-12-09	December	12	2014
2015-02-18	February	2	2015

Data Analysis

How features of a home influence its sale price

Specifically, we will be using
Square-Footage of all Living Areas
# of Bedrooms
# of Bathrooms

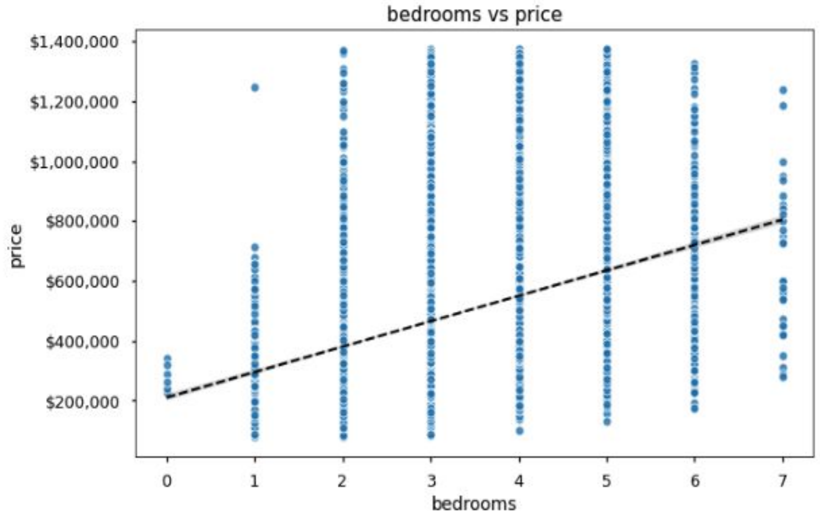
House Price Distribution Plot

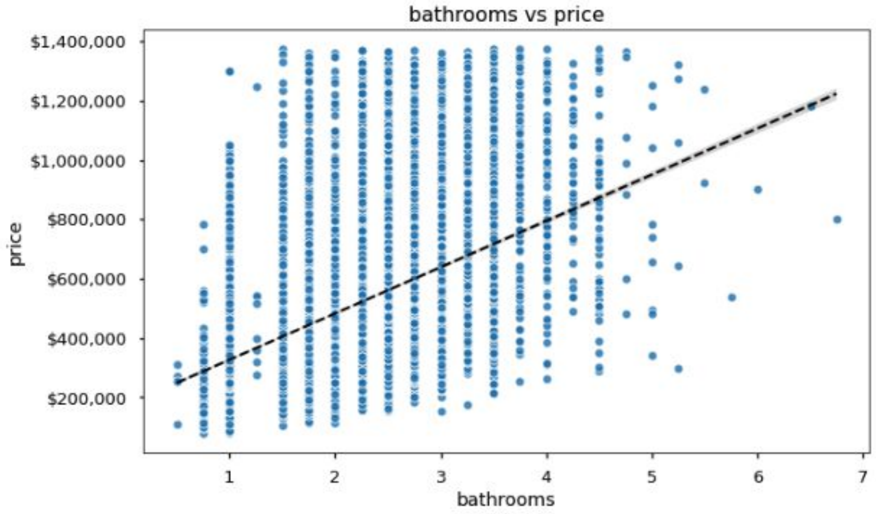


Relation between sqft_living and Price

Positive relationship between sqft-living and price

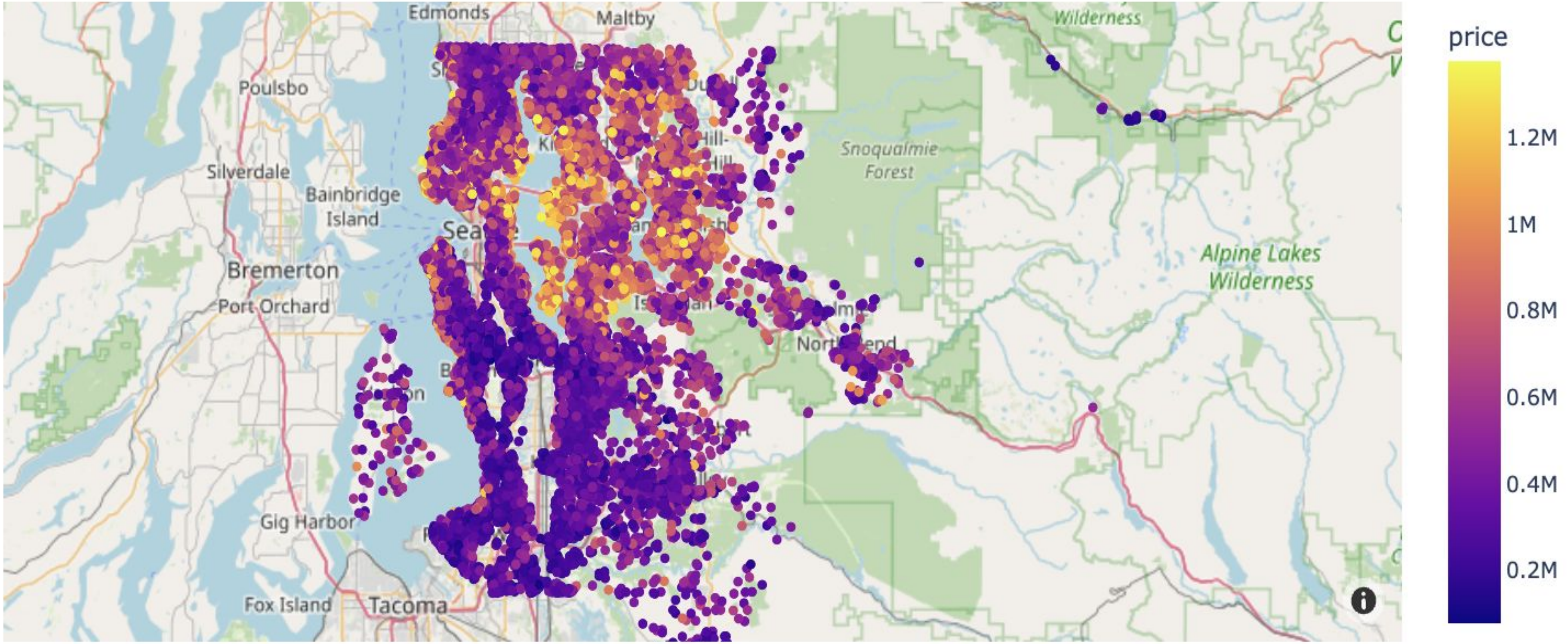
Relationship Between Features vs Price





How would you describe the relationship between price and features shown in graph.

King County , WA House Sales (2014-2015)



Modeling

Let's create a Linear Regression model with sci-kit learn to determine the effect of features on price

Supervised Learning
Features and Target

Predict the Sales Price of a home …

Use this data to… **Predict this value or class**

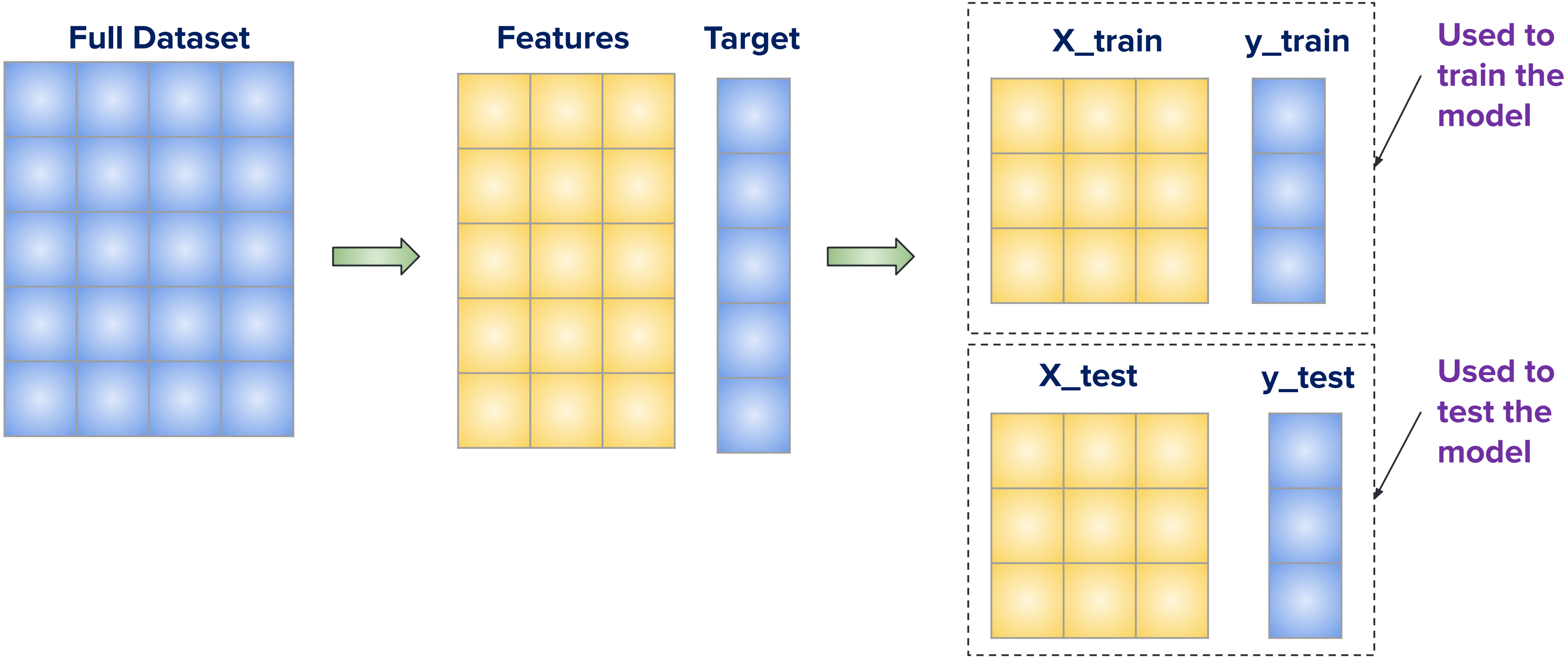
Square Footage	Number of Bedrooms	Number of Bathrooms	Sales Price
2000	3	2	500,000
2500	4	3	650,000

Features
Independent Variables
(X)

Target
Dependent Variable
(y)

Supervised Learning

Features and Target - Train Test Split



Model Training

```
# Create our X & y using bedrooms,bathrooms, sqft-living
use_cols = ['bedrooms','bathrooms','sqft_living','yr_built']
#"sqft_above","grade","sqft_living"]
X = df[use_cols].copy()
y = df['price'].copy()

## Train test split (random-state 321, test_size=0.25)
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state=321)
X_train
```

	bedrooms	bathrooms	sqft_living	yr_built
id				
3336001946	2	1.0000	900	1951
2925059260	5	2.5000	3000	1966
1433290010	3	2.2500	1960	1984
6668900020	4	2.0000	1370	1947

Linear Regression

Glass Box because:

- Calculates values (coefficients) for each feature, which we can access, inspect, and use to calculate prediction ourselves.

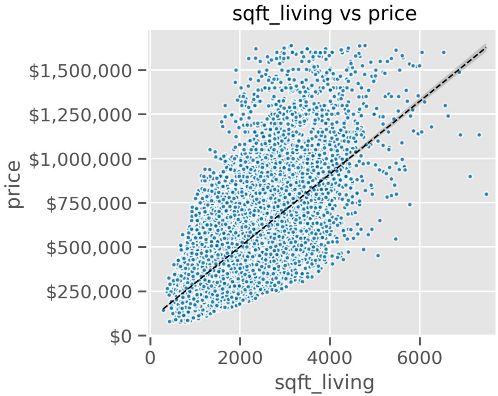
What you are trying to predict

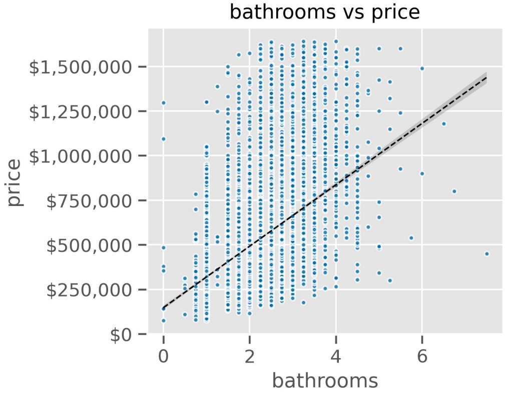
Your feature(s)

Where your line crosses the y-axis

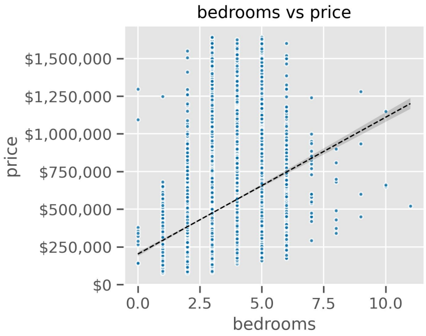
$$y = mx + b$$

Rate of Change/Slope





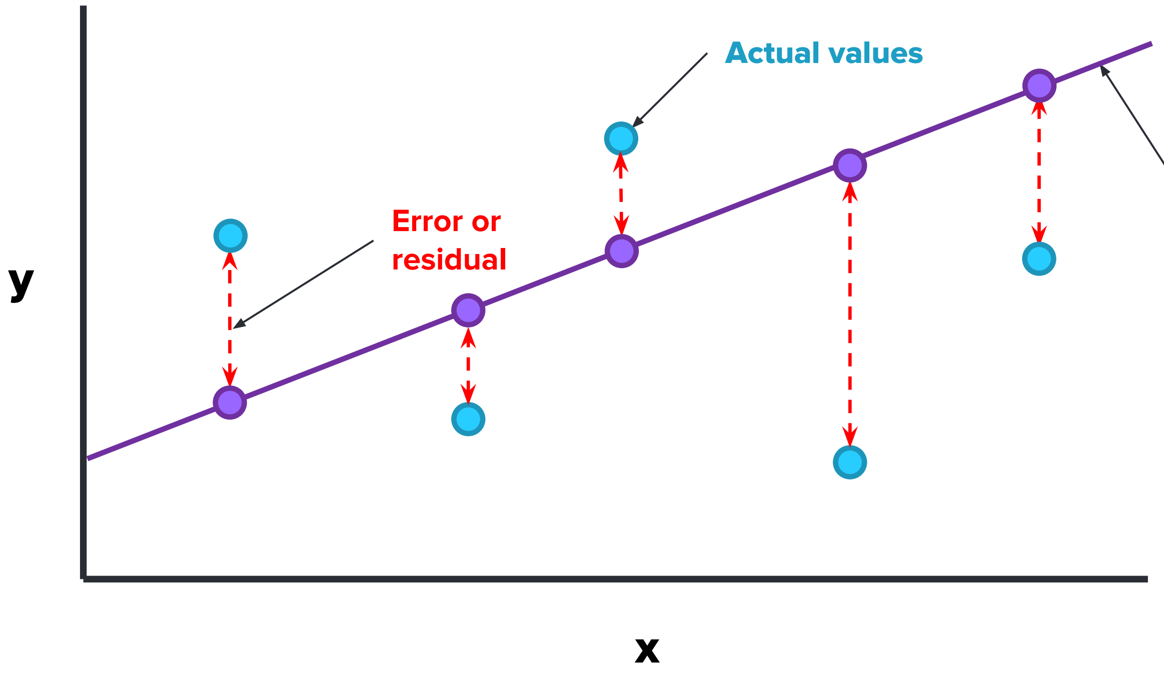
$$y = \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_0$$



Regression Evaluation Metrics

- **Mean Absolute Error (MAE)**
- **Mean Squared Error (MSE)**
- **Root Mean Squared Error (RMSE)**
- **R-Squared (R2) or (R^2) or (R^2)**

Regression Evaluation Metrics



Predicted values or
trend line

Error:
(Predicted - True Value)
can be positive or negative.

Regression Error Metrics

	MAE Mean Absolute Error	MSE Mean Squared Error	RMSE Root Mean Squared Error
Penalizes Large Errors	No	Yes	Yes
Benefits	Same units of measure as the target	NOT the same units of measure as the target	Same units of measure as the target
Where to use *(Trying to giving you idea, this is not fixed because it depends on business problem)*	MAE does not punish large error so we use it oftenly when you really don't bother about the large errors .	It is useful when your dataset contains lot of noise because it punishes the large errors.	RMSE assigns a higher weight to larger errors. This is much more useful when large errors are present and they drastically affect the model's performance.

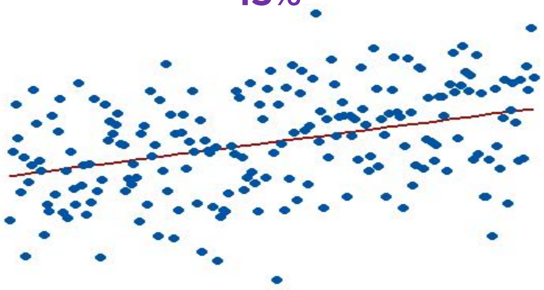
R-Squared R^2

(The Coefficient of Determination)

- **R-squared (R^2) represents the proportion of the variance for a dependent variable that's explained by an independent variable or variables in a regression model.**

- **R-squared explains to what extent the variance of one variable(independent variable) explains the variance of the second variable(dependent variable). So, if the R^2 of a model is 0.60, then approximately more than half of the observed variation can be explained by the model's inputs.**

R-Squared: 15%



R-Squared: 85%

Model Testing

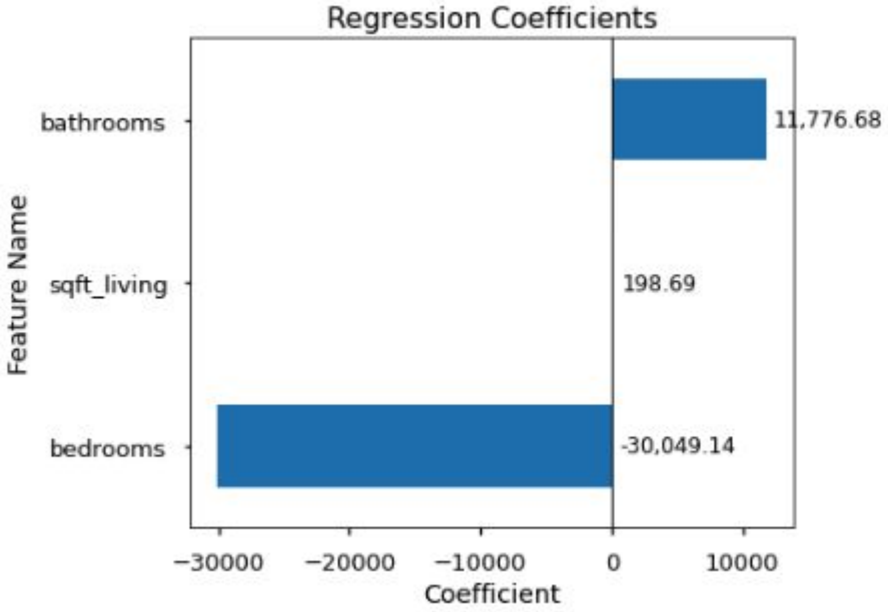
```
reg = LinearRegression()
reg.fit(X_train,y_train)
evaluate_regression(reg, X_train, y_train, X_test, y_test)
```

```
Training Data:  R^2 = 0.43      RMSE = 178,452.79       MAE = 138,453.63
Test Data:      R^2 = 0.44      RMSE = 177,104.24       MAE = 136,839.84
```

$$y = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_0$$

```
## Get the coefficents from the model using our custom
coeffs = get_coeffs(reg,X_train)
coeffs
```

```
bedrooms       -30,049.1422
bathrooms       11,776.6798
sqft_living        198.6927
Intercept      170,535.1411
dtype: float64
```

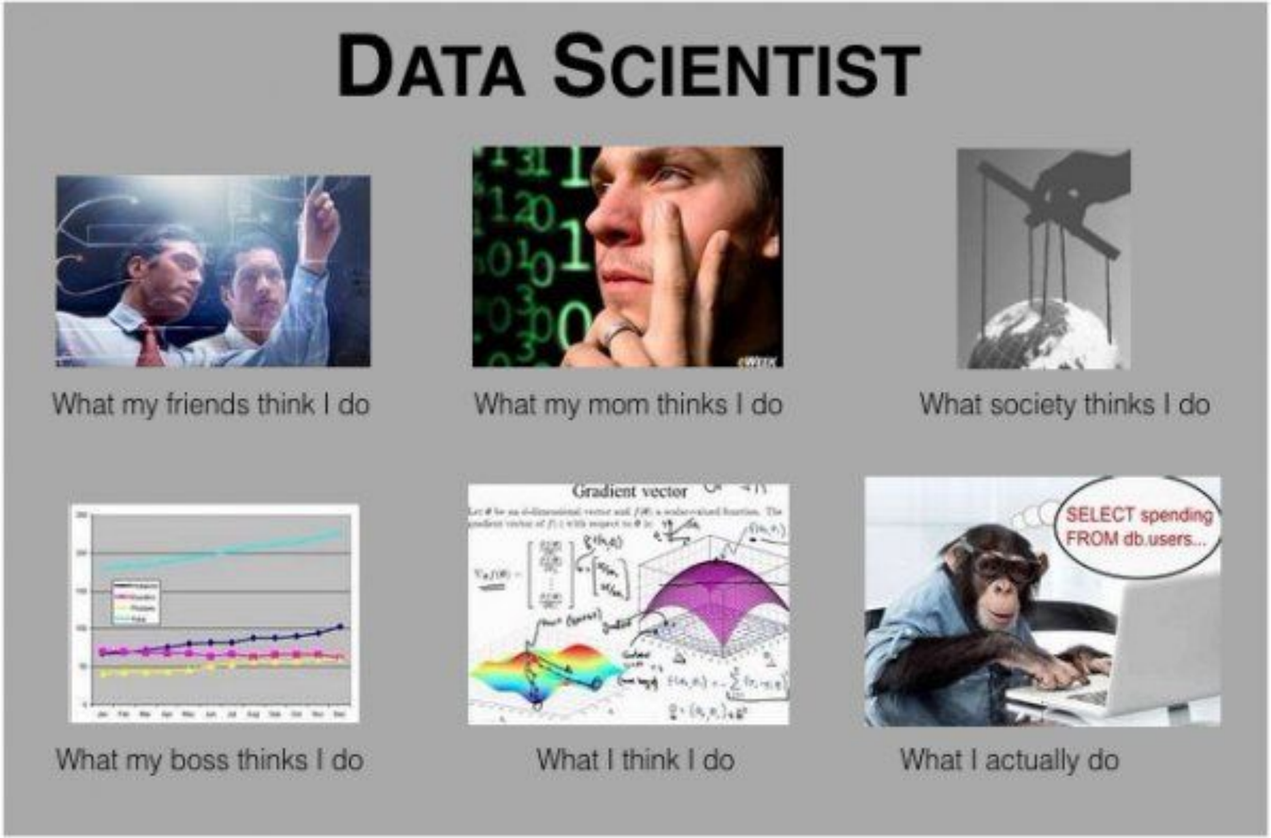


- **Each coefficient tells us the effect of increasing the values in that column by 1 unit.**
- According to our model, we can determine a home's price using the following results:
  - The model assumed a default/starting house price was $170,535.1411 (the intercept)
  - For each additional bedrooms, subtract $-30,049.1422
  - For each batrhoom, add $11,776.6798
  - For each square foot of living space, add $198.6927

Business Recommendation ,Next Steps and Presentation

- Location has high influence on house price
- Square foot living has strong correlation with price
- Possible association between Bedrooms and Square foot living as more bedrooms lower the house price based on square foot
- Including other features and tuning the model could increase the regression metrics

[Link to Github Repository]



Questions?