# Data Science & ML Course

## Lesson #17 K-Nearest Neighbors

Ivanovitch Silva

November, 2018

# Update from repository

```
git clone https://github.com/ivanovitchm/datascience2machinelearning.git
```

Or ….

```
git pull
```

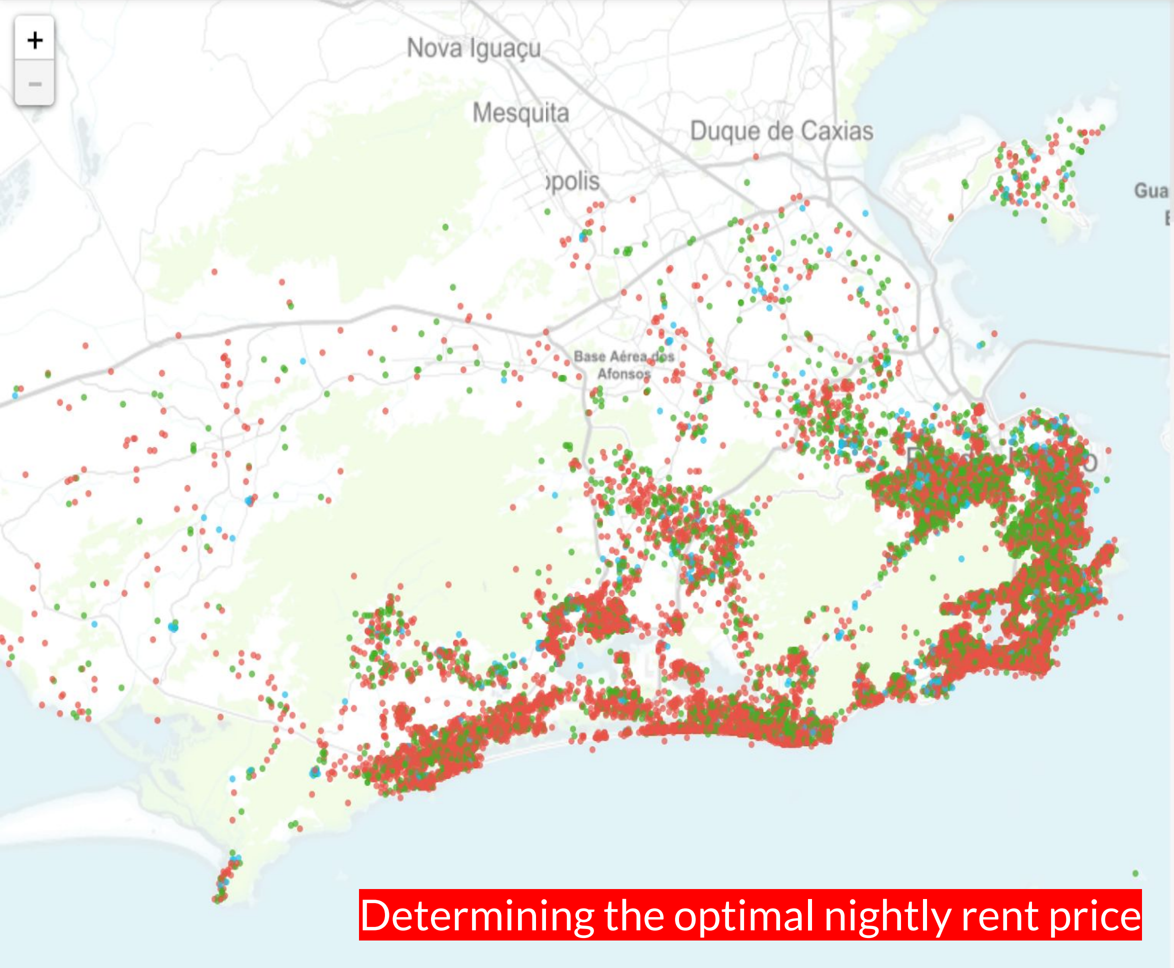
# Agenda

- Univariate KNN
  - Euclidean distance for univariate
  - Function to make predictions
  - Error metrics
- Multivariate KNN
  - Normalize columns
  - Euclidean distance for multivariate
- Hyperparameter optimization
- Cross-Validation

**Inside Airbnb**
Adding data to the debate

About | Behind | Get the Data

Like 0 | Share | Tweet




Determining the optimal nightly rent price

# Rio de Janeiro

Filter by:

Rio de Janeiro

**32,469**

out of **32,469** listings (100%)

About Airbnb in Rio de Janeiro

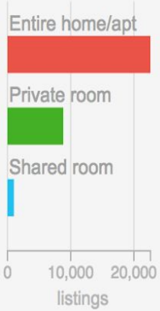
How is Airbnb really being used in and affecting your neighbourhoods?

## Room Type

☐ Only entire homes/apartments

Airbnb hosts can list entire homes/apartments, private or shared rooms.

Depending on the room type and activity, an airbnb listing could be more like a hotel, disruptive for neighbours, taking away housing, and illegal.




**69.7%**
entire homes/apartments

**R$682**
price/night

**22,631 (69.7%)**
entire home/apartments

**8,822 (27.2%)**
private rooms

**1,016 (3.1%)**
shared rooms

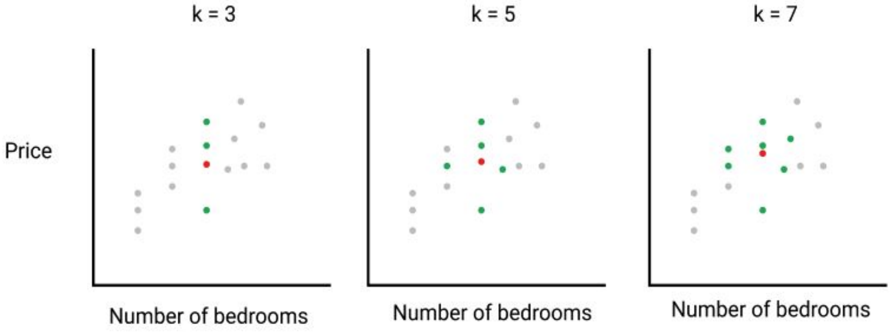
- host_response_rate: the response rate of the host
- host_acceptance_rate: number of requests to the host that convert to rentals
- host_listings_count: number of other listings the host has
- latitude: latitude dimension of the geographic coordinates
- longitude: longitude part of the coordinates
- city: the city the living space resides
- zipcode: the zip code the living space resides
- state: the state the living space resides
- accommodates: the number of guests the rental can accommodate
- room_type: the type of living space (Private room, Shared room or Entire home/apt
- bedrooms: number of bedrooms included in the rental
- bathrooms: number of bathrooms included in the rental
- beds: number of beds included in the rental
- price: nightly price for the rental
- cleaning_fee: additional fee used for cleaning the living space after the guest leaves
- security_deposit: refundable security deposit, in case of damages
- minimum_nights: minimum number of nights a guest can stay for the rental
- maximum_nightss: maximum number of nights a guest can stay for the rental
- number_of_reviews: number of reviews that previous guests have left

96
cols

```
total 112M
-rw-r--r-- 1 root root 112M Sep 29 21:46 listings.csv
drwxr-xr-x 2 root root   1M Sep 27 20:32 sample_data
```

# K-Nearest Neighbors

Select the number of similar listings, **k**, you want to compare with.




For this example, we'll use 3 for our **k** value.

For each listing, calculate how similar it is to our unpriced listing.

dataset

| bedrooms | price |
|---|---|
| 1 | 160 |
| 3 | 350 |
| 1 | 60 |
| 1 | 95 |
| 1 | 50 |

compare each listing with

our unpriced listing

| bedrooms | price |
|---|---|
| 1 | ? |

Rank each listing by the similarity metric and select the first **k** listings.

dataset (ordered by similarity)

| bedrooms | price | similarity |
|---|---|---|
| 1 | 160 | 0 |
| 1 | 60 | 0 |
| 1 | 95 | 0 |
| 1 | 50 | 0 |
| 3 | 350 | 2 |

our unpriced listing

| bedrooms | price |
|---|---|
| 1 | ? |

Calculate the mean list price for the **k** similar listings and use as our list price.

dataset (ordered by similarity)

| bedrooms | price |
|---|---|
| 1 | 160 |
| 1 | 60 |
| 1 | 95 |

mean price

105

# Euclidean distance

$$d = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)^2}$$

| index | host_listings_count | accommodates | bedrooms | bathrooms | beds |
|---|---|---|---|---|---|
| 0 | 26 | 4 | 1 | 1 | 2 |
| 1 | 1 | 6 | 3 | 3 | 3 |

$(q_1 - p_1) + (q_2 - p_2) + \ldots + (q_n - p_n)$

differences: $(26 - 1) + (4 - 6) + (1 - 3) + (1 - 3) + (2 - 3)$

$(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)^2$

squared differences: $(25)^2 + (-2)^2 + (-2)^2 + (-2)^2 + (-1)^2$

$\sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)^2}$

Euclidean distance

$$= \sqrt{625 + 4 + 4 + 4 + 1}$$

$$= \sqrt{638}$$

$$= 25.258661$$

# Euclidean distance - example

| | accommodates |
|---|---|
| our listing | 8 |

## Univariate case

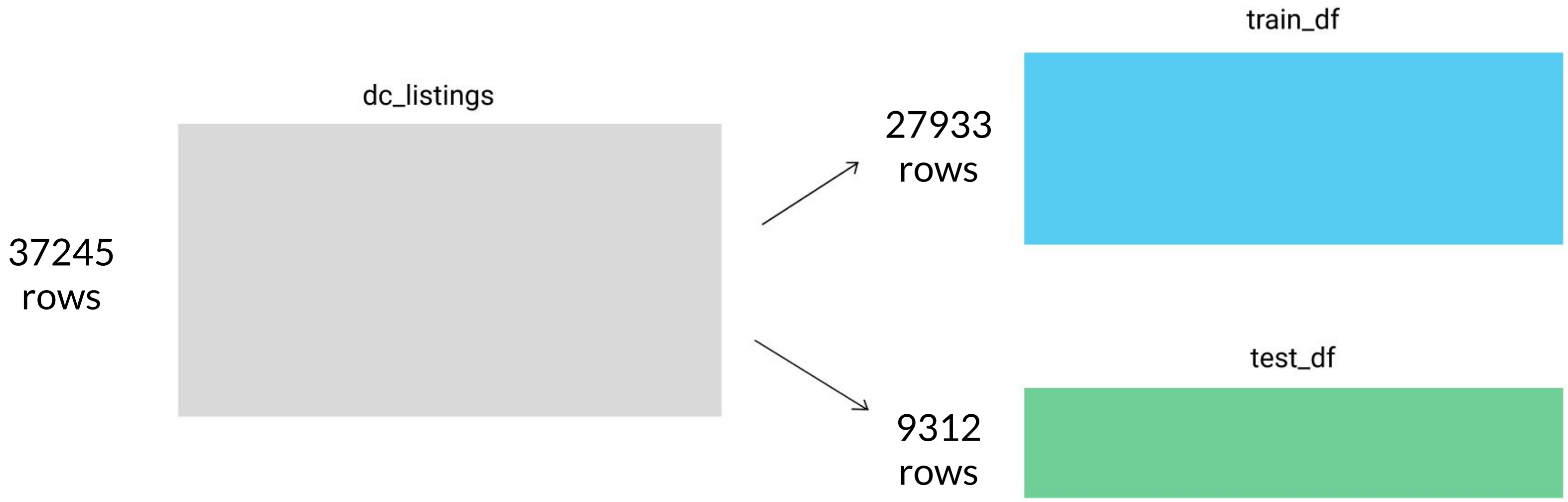
$$d = \sqrt{(q_1 - p_1)^2}$$

$$d = |q_1 - p_1|$$

| | index | accommodates | distance |
|---|---|---|---|
| dc_listings | 0 | 4 | $(4 - 8)^2$ |
| | 1 | 6 | $(6 - 8)^2$ |
| | 2 | 1 | $(1 - 8)^2$ |
| | 3 | 2 | $(2 - 8)^2$ |

# Testing quality of predictions

## Machine Learning Model



# Error metrics

## Mean Absolute Error

$$MAE = \frac{|actual_1 - predicted_1| + |actual_2 - predicted_2| + \ \ldots + |actual_n - predicted_n|}{n}$$

## Mean Squared Error

$$MSE = \frac{(actual_1 - predicted_1)^2 + (actual_2 - predicted_2)^2 + \ \ldots + (actual_n - predicted_n)^2}{n}$$

## Root Mean Squared Error

$$RMSE = \sqrt{MSE}$$

https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d

# Improve the accuracy (multivariate knn)

- Increase the number of attributes the model uses to calculate similarity when ranking the closest neighbors
- Increase k, the number of nearby neighbors the model uses when computing the prediction
- We'll focus on more robust techniques for testing a machine learning model's accuracy, namely, cross-validation

# Improve the accuracy (multivariate knn)

When selecting more attributes to use in the model, we need to watch out for columns that don't work well with the distance equation.

- non-numerical values (e.g. city or state)
    - Euclidean distance equation expects numerical values
- missing values
    - distance equation expects a value for each observation and attribute
- non-ordinal values (e.g. latitude or longitude)
    - ranking by Euclidean distance doesn't make sense if all attributes aren't ordinal

# Removing features

- room_type: e.g. **Private room**
- city: e.g. **Washington**
- state: e.g. **DC**

- host_response_rate
- host_acceptance_rate
- host_listings_count

- latitude: e.g. **38.913458**
- longitude: e.g. **-77.031**
- zipcode: e.g. **20009**

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 37245 entries, 3629 to 33003
Data columns (total 10 columns):
accommodates          37245 non-null int64
bathrooms             37168 non-null float64
bedrooms              37219 non-null float64
beds                  37188 non-null float64
price                 37245 non-null float64
security_deposit      18540 non-null object
cleaning_fee          23159 non-null object
minimum_nights        37245 non-null int64
maximum_nights        37245 non-null int64
number_of_reviews     37245 non-null int64
dtypes: float64(4), int64(4), object(2)
memory usage: 3.1+ MB
```

# Normalize columns

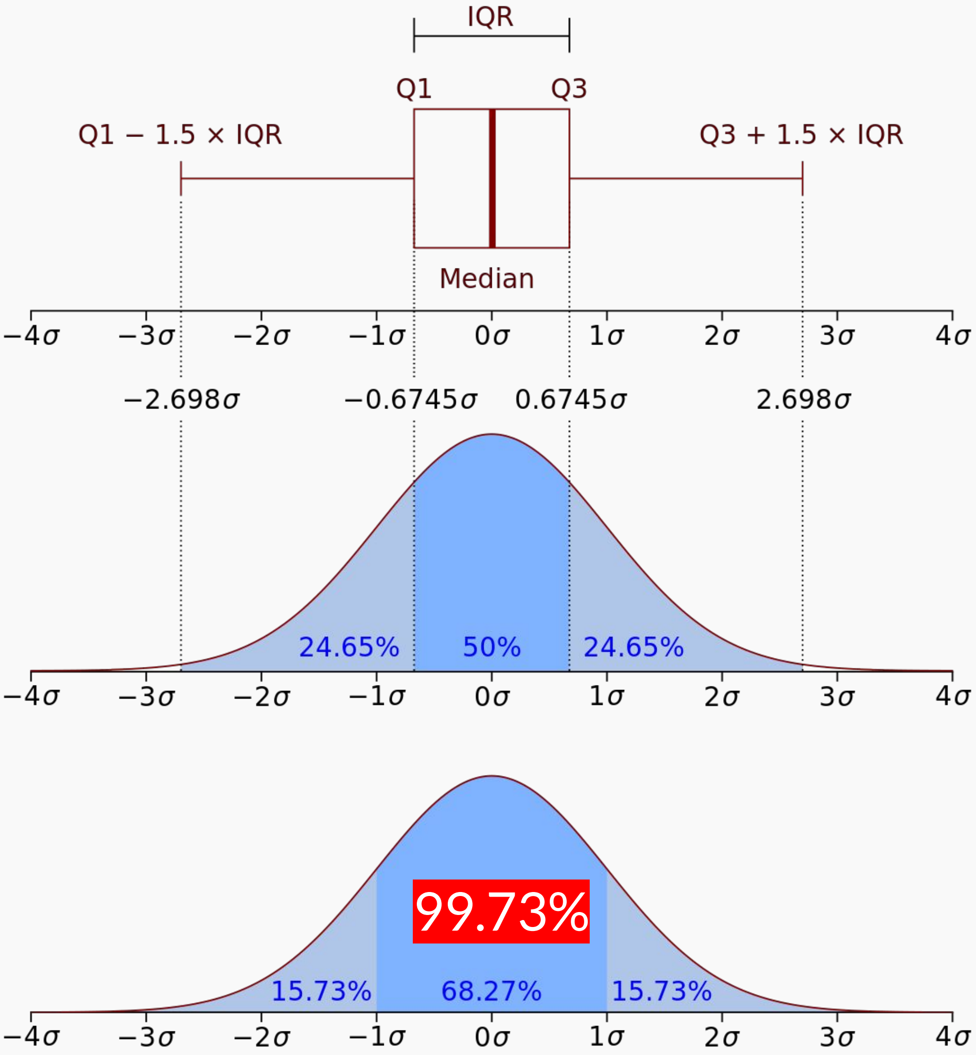
| accommodates | bathrooms | bedrooms | beds | price | minimum_nights | maximum_nights | number_of_reviews |
|---|---|---|---|---|---|---|---|
| 8 | 1.0 | 1.0 | 1.0 | 200.0 | 2 | 60 | 0 |
| 6 | 2.0 | 3.0 | 4.0 | 901.0 | 10 | 27 | 0 |
| 2 | 1.0 | 1.0 | 1.0 | 229.0 | 2 | 50 | 24 |
| 2 | 1.0 | 1.0 | 2.0 | 200.0 | 3 | 1125 | 0 |
| 2 | 2.0 | 1.0 | 1.0 | 75.0 | 2 | 30 | 2 |

| accommodates | bathrooms | bedrooms | beds | price | minimum_nights | maximum_nights | number_of_reviews |
|---|---|---|---|---|---|---|---|
| 1.455750 | -0.67557 | -0.597149 | -0.779645 | 200.0 | -0.121917 | -0.008480 | -0.337027 |
| 0.691698 | 0.27004 | 1.229023 | 0.665006 | 901.0 | 0.254259 | -0.008483 | -0.337027 |
| -0.836405 | -0.67557 | -0.597149 | -0.779645 | 229.0 | -0.121917 | -0.008481 | 0.981934 |
| -0.836405 | -0.67557 | -0.597149 | -0.298095 | 200.0 | -0.074895 | -0.008400 | -0.337027 |
| -0.836405 | 0.27004 | -0.597149 | -0.779645 | 75.0 | -0.121917 | -0.008482 | -0.227114 |

# Normalize columns (outliers)

```
normalized_listings.describe()
```

| | accommodates | bathrooms | bedrooms | beds | price | minimum_nights | maximum_nights | number_of_reviews |
|---|---|---|---|---|---|---|---|---|
| **count** | 3.712900e+04 | 3.712900e+04 | 3.712900e+04 | 3.712900e+04 | 37129.000000 | 3.712900e+04 | 3.712900e+04 | 3.712900e+04 |
| **mean** | -1.314721e-16 | -4.592913e-18 | -4.822558e-17 | -4.478090e-17 | 645.284872 | -2.066811e-17 | 7.654855e-19 | 2.387358e-17 |
| **std** | 1.000000e+00 | 1.000000e+00 | 1.000000e+00 | 1.000000e+00 | 1658.580291 | 1.000000e+00 | 1.000000e+00 | 1.000000e+00 |
| **min** | -1.218415e+00 | -1.621158e+00 | -1.510215e+00 | -1.261178e+00 | 0.000000 | -1.689371e-01 | -8.484452e-03 | -3.370226e-01 |
| **25%** | -8.363938e-01 | -6.755609e-01 | -5.971412e-01 | -7.796344e-01 | 150.000000 | -1.689371e-01 | -8.482279e-03 | -3.370226e-01 |
| **50%** | -7.235234e-02 | -6.755609e-01 | -5.971412e-01 | -2.980906e-01 | 296.000000 | -1.219157e-01 | -8.400222e-03 | -3.370226e-01 |
| **75%** | 3.096684e-01 | 2.700359e-01 | 3.159325e-01 | 1.834533e-01 | 626.000000 | -2.787293e-02 | -8.400222e-03 | -1.721546e-01 |
| **max** | 5.952288e+01 | 1.729078e+01 | 3.957810e+01 | 2.281602e+01 | 41719.000000 | 5.733824e+01 | 1.609190e+02 | 1.647951e+01 |


IQR
Q1
Q3
Q1 − 1.5 × IQR
Q3 + 1.5 × IQR
Median
−4σ −3σ −2σ −1σ 0σ 1σ 2σ 3σ 4σ
−2.698σ
−0.6745σ
0.6745σ
2.698σ
24.65%
50%
24.65%
−4σ −3σ −2σ −1σ 0σ 1σ 2σ 3σ 4σ
99.73%
15.73%
68.27%
15.73%
−4σ −3σ −2σ −1σ 0σ 1σ 2σ 3σ 4σ

# Euclidean distance for multivariate case

| accommodates | bathrooms |
|---|---|
| -0.596544 | -0.439151 |
| -0.596544 | 0.412923 |

| | | |
|---|---|---|
| $(q_1 - p_1) + (q_2 - p_2) + \ldots + (q_n - p_n)$ | differences | $(-0.596544 + 0.596544) + (-0.439151 - 0.412923)$ |
| $(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)^2$ | squared differences | $(0)^2 + (-0.852074)^2$ |
| $\sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \ldots + (q_n - p_n)^2}$ | Euclidean distance | $= \sqrt{0 + 0.72603}$ <br> $= 0.852074$ |

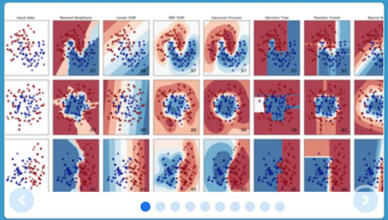
# Introduction to scikit-learn

scikit learn

Home | Installation | Documentation | Examples

Google Custom Search | Search

Fork me on GitHub



## scikit-learn

*Machine Learning in Python*

- Simple and efficient tools for data mining and data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

### Classification

Identifying to which category an object belongs to.

**Applications**: Spam detection, Image recognition.
**Algorithms**: SVM, nearest neighbors, random forest, …

— Examples

### Regression

Predicting a continuous-valued attribute associated with an object.

**Applications**: Drug response, Stock prices.
**Algorithms**: SVR, ridge regression, Lasso, …

— Examples

### Clustering

Automatic grouping of similar objects into sets.

**Applications**: Customer segmentation, Grouping experiment outcomes
**Algorithms**: k-Means, spectral clustering, mean-shift, …

— Examples

### Dimensionality reduction

Reducing the number of random variables to consider.

**Applications**: Visualization, Increased efficiency
**Algorithms**: PCA, feature selection, non-negative matrix factorization.

— Examples

### Model selection

Comparing, validating and choosing parameters and models.

**Goal**: Improved accuracy via parameter tuning
**Modules**: grid search, cross validation, metrics.

— Examples

### Preprocessing

Feature extraction and normalization.

**Application**: Transforming input data such as text for use with machine learning algorithms.
**Modules**: preprocessing, feature extraction.

— Examples

# Scikit-learn workflow

The scikit-learn workflow consists of 4 main steps:

- instantiate the specific machine learning model you want to use
- fit the model to the training data
- use the model to make predictions
- evaluate the accuracy of the predictions

# Scikit-learn workflow

```
# features
train_columns = ['accommodates', 'bathrooms']

# instantiate a knn object
knn = KNeighborsRegressor(n_neighbors=5, algorithm='brute', metric='euclidean')

# train the model
knn.fit(train_df[train_columns], train_df['price'])

# predict
predictions = knn.predict(test_df[train_columns])

# evaluate
rmse = np.sqrt(mean_squared_error(predictions,test_df.price))
```

# Improve the accuracy (multivariate knn)

- We'll focus on the impact of increasing k, the number of nearby neighbors the model uses to make predictions.

# Hyperparameters

- When we **vary the features** that are used in the model, we're **affecting the data** that the model uses.
- On the other hand, **varying the k value affects the behavior of the model independently of the actual data** that's used when making predictions.
- Values that affect the behavior and performance of a model that are unrelated to the data that's used are referred to as hyperparameters.

# Hyperparameter Optimization

A simple but common hyperparameter optimization technique is known as grid search:

- selecting a subset of the possible hyperparameter values,
- training a model using each of these hyperparameter values,
- evaluating each model's performance,
- selecting the hyperparameter value that resulted in the lowest error value.

# Improve the accuracy (multivariate knn)

- We can focus on more robust techniques for testing a machine learning model's accuracy

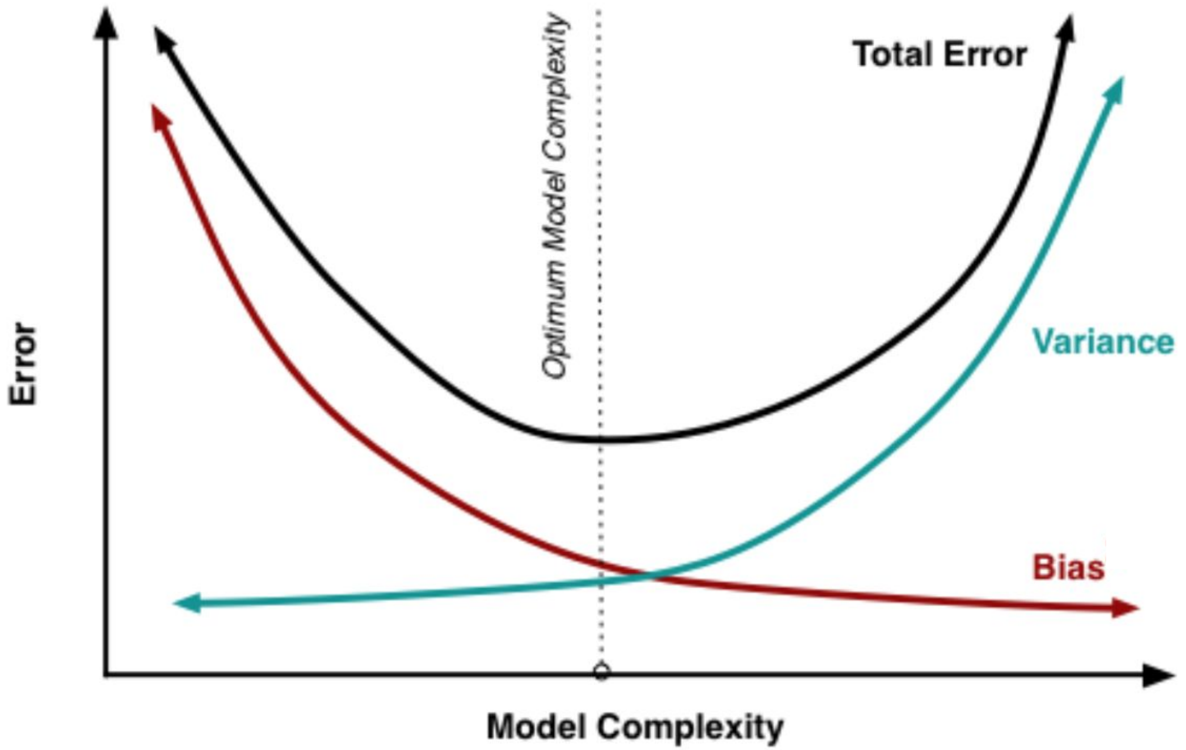
# Holdout Validation


Holdout validation is actually a specific example of a larger class of validation techniques called **k-fold cross-validation.**

# K-Fold Cross Validation

| | | | | | |
|---|---|---|---|---|---|
| | Test | Train | Train | Train | Train |
| | Train | Test | Train | Train | Train |
| | Train | Train | Test | Train | Train |
| | Train | Train | Train | Test | Train |
| | Train | Train | Train | Train | Test |
| Errors | 120.55 | 122.11 | 125.91 | 123.41 | 122.81 |
| Mean Error | | | 122.96 | | |

# Bias-Variance Tradeoff




MacBook Pro