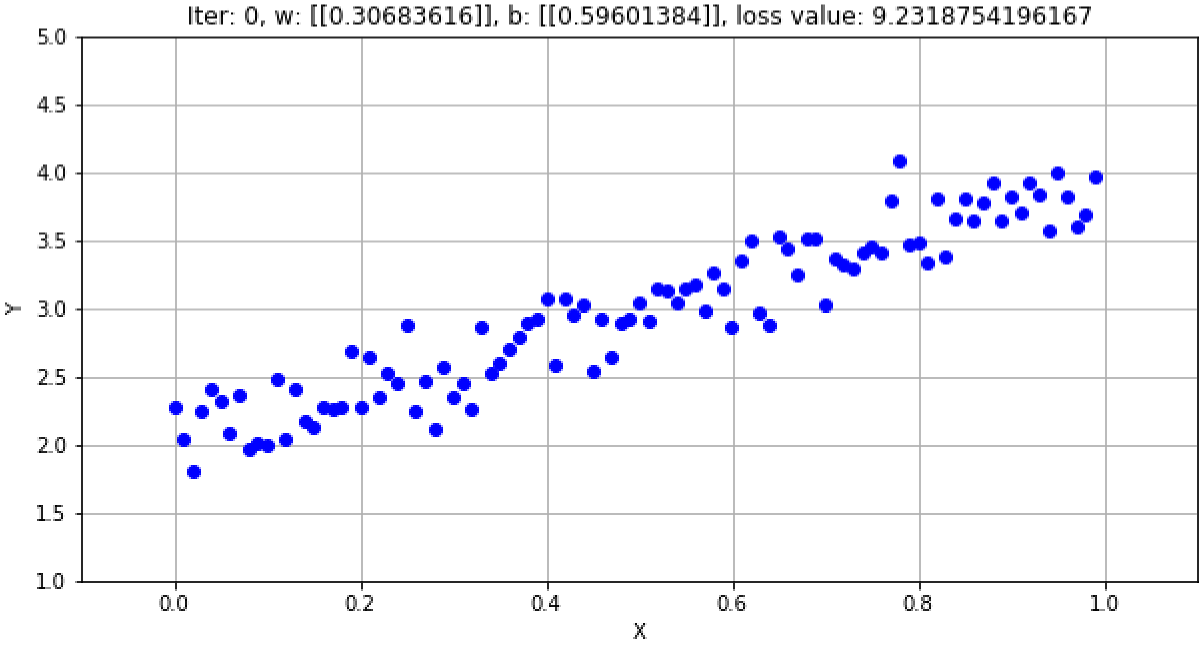
ML with Big Data

Professor Ernesto Lee

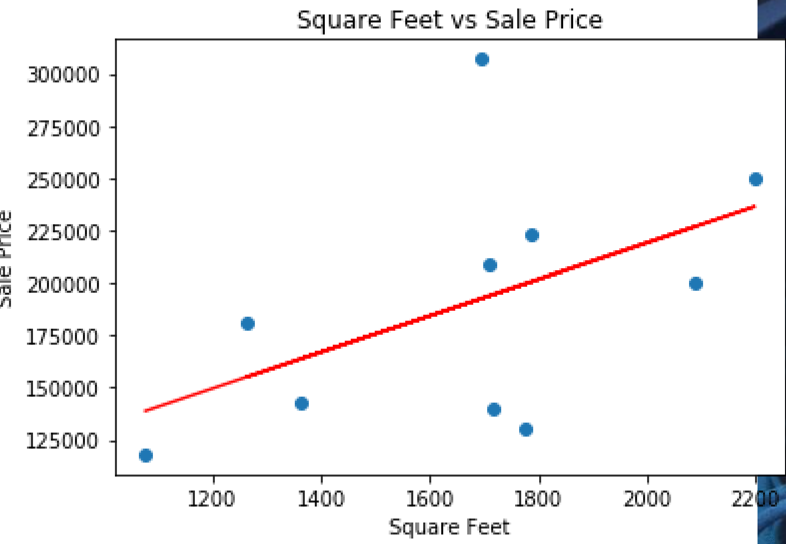
HTTPS://BIT.LY/MLTRAIN

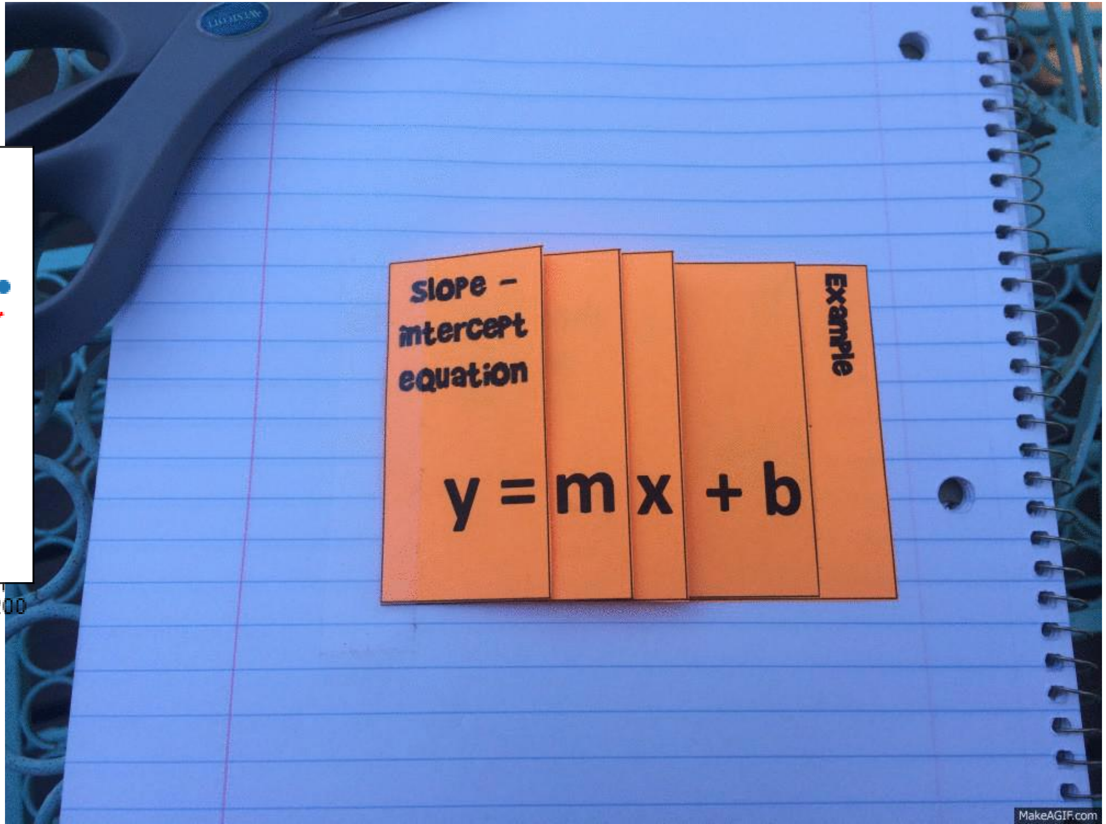
Go here for your LAB ENVIRONMENT

Linear Relationships



Regression Algorithm





Linear Regression with Scikit-Learn

Import Your Libraries

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn import metrics
```

Pull in Your Data

You can access your data here:
https://bit.ly/21homes

Or here:

https://raw.githubusercontent.com/fenago/pythonml/main/data/HousePrice.csv

----

```
df =
pd.read_csv('https://raw.githubusercontent.com/fenago/pythonml/main/data/HousePrice.csv')
```

Putting the Columns You Want In Your Data

```
df2 = df[["open", "close"]] #open and close are the columns
```

Make sure you have the columns you want and need!

Understand Your Data with Shape

`df.shape`

```
df.shape
```

```
(1460, 2)
```

Understand Your Data: HEAD

`df.head()`

```
df.head()
```

	SquareFeet	SalePrice
0	1710	208500
1	1262	181500
2	1786	223500
3	1717	140000
4	2198	250000

Understand Your Data: Descriptive Statistics

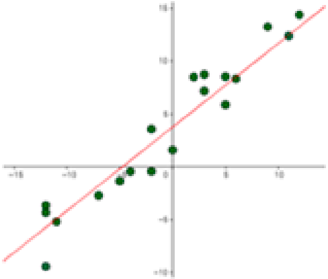
`df.describe()`

```
[8] df.describe()
```

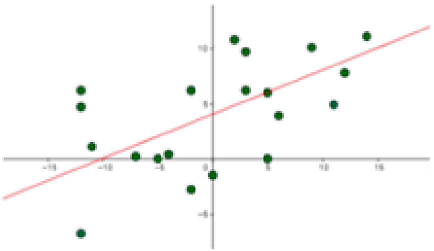
	SquareFeet	SalePrice
count	1460.000000	1460.000000
mean	1515.463699	180921.195890
std	525.480383	79442.502883
min	334.000000	34900.000000
25%	1129.500000	129975.000000
50%	1464.000000	163000.000000
75%	1776.750000	214000.000000
max	5642.000000	755000.000000

Find Your Correlations In your Datasets

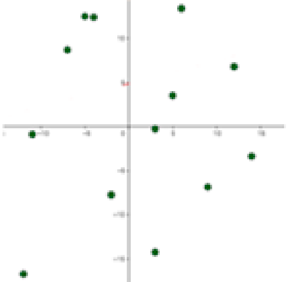
```
# correlation between 2 Specific Columns

print(df['SquareFeet'].corr(df['SalesPrice']))



# pair-wise correlation between all columns


print(df.corr())
```



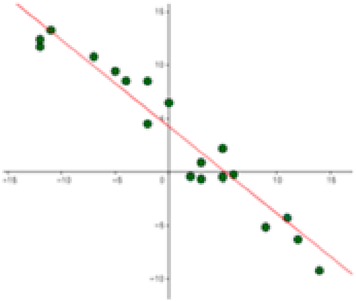
Strong Positive Correlation



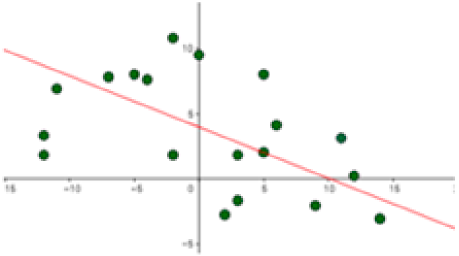
Weak Positive Correlation



No Correlation



Strong Negative Correlation



Weak Negative Correlation

HeatMap

```
# Correlation between different variables
corr = df.corr()
# Set up the matplotlib plot configuration
f, ax = plt.subplots(figsize=(12, 10))
# Generate a mask for upper traingle
mask = np.triu(np.ones_like(corr, dtype=bool))
# Configure a custom diverging colormap
cmap = sns.diverging_palette(230, 20, as_cmap=True)
# Draw the heatmap
sns.heatmap(corr, annot=True, mask = mask, cmap=cmap)
```

Visualize Your Data: PLOT

```
df.plot(x='SquareFeet', y='SalePrice', style='*')

plt.title('Square Feet vs Sale Price')

plt.xlabel('Square Feet')

plt.ylabel('Sale Price')

plt.show()
```

Prepare Your Data: Split Into Training and Test Sets

```
X = df.iloc[:, :-1].values

y = df.iloc[:, 1].values
```

Train Test Split

```
X_train, X_test, y_train, y_test = train_test_split(X,
y, test_size=0.2, random_state=0)
```

Run the Model

```
def get_cv_scores(model):
    scores = cross_val_score(model, X_train, y_train, cv=10, scoring='r2')

    print('CV Mean: ', np.mean(scores))
    print('STD: ', np.std(scores))
    print('\n')

lr = LinearRegression().fit(X_train, y_train)
get_cv_scores(lr)
```

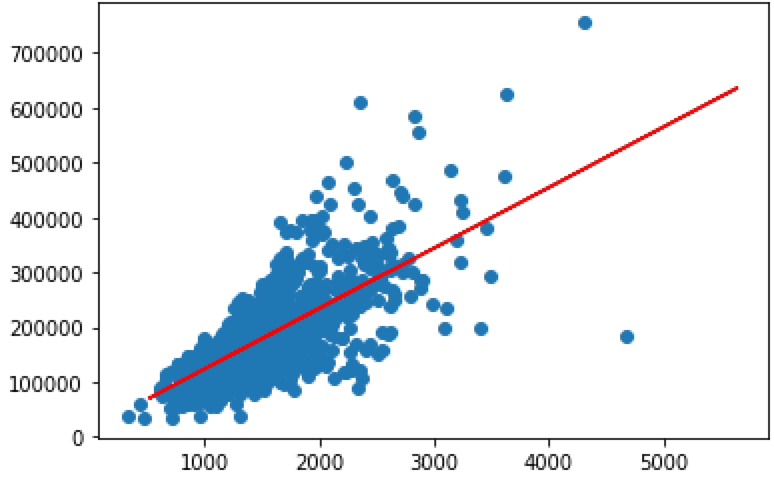
View the Coefficients

```
print(lr.intercept_)

print(lr.coef_)
```

```
print(lr.intercept_)
print(lr.coef_)
```

```
13330.293444921088
[110.26434426]
```

Predictions



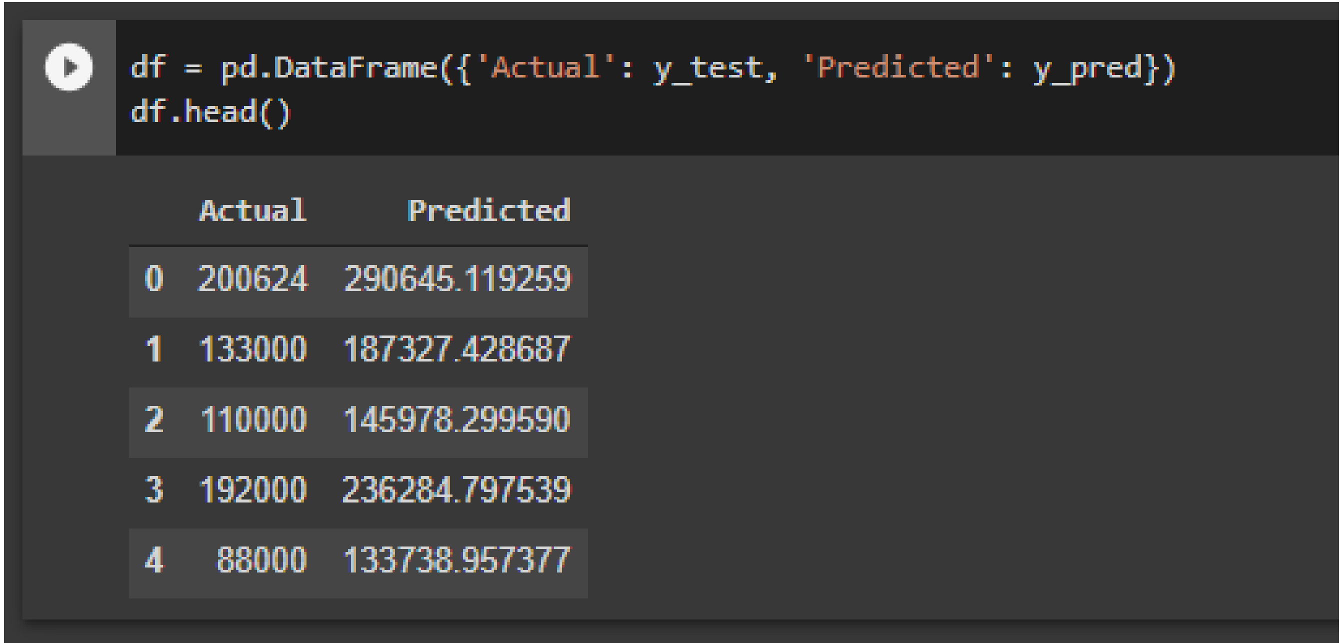

```
y_pred = lr.predict(X_test)


plt.scatter(X_train, y_train)

plt.plot(X_test, y_pred, color='red')

plt.show()
```

Predict with NEW unseen Data

```
lr.predict([[2515]])
```

Evaluate Predicted Values from Actuals

```
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})

df.head()
```



```
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df.head()
```

	Actual	Predicted
0	200624	290645.119259
1	133000	187327.428687
2	110000	145978.299590
3	192000	236284.797539
4	88000	133738.957377

R-Squared (R2)

```
df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
df.head()


print('R-Squared:',metrics.r2_score(df['Actual'],df['Predicted']))
------
```

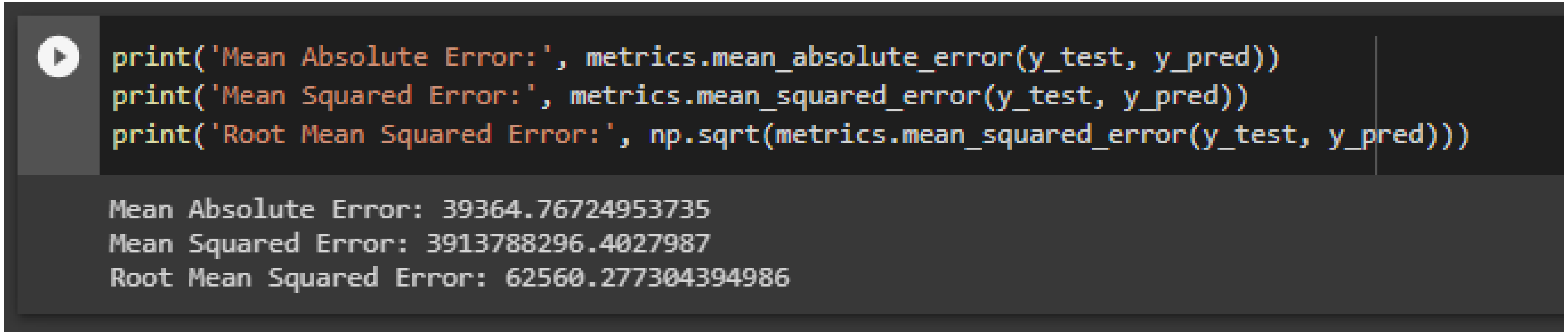
Summary Definition

Define R-Squared: Coefficient of determination means a statistical measurement of the correlation between two variables.

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.r2_score.html

Evaluate Your Model

```
print('Mean Absolute Error:',
metrics.mean_absolute_error(y_test, y_pred))

print('Mean Squared Error:',
metrics.mean_squared_error(y_test, y_pred))

print('Root Mean Squared Error:',
np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```



https://scikit-learn.org/

https://github.com/scikit-learn/scikit-learn

Tuning Hyperparameters with Simple Linear Regression

https://bit.ly/ucidata

Hint: To create a new dataframe with selected columns - do this:

df2 = df[["open", "close"]] #open and close are the colomns

Lab 1

Read in this DATA:

```
pd.read_csv('https://raw.githubusercontent.com/fenago/pythonml/main/data/poverty.txt',sep="\t")
```

Apply what you have learned to create Simple Linear Models.

This dataset of size $n = 51$ are for the 50 states and the District of Columbia in the United States (poverty.txt). The variables are y = year 2002 birth rate per 1000 females 15 to 17 years old and x = poverty rate, which is the percent of the state's population living in households with incomes below the federally defined poverty level. (Data source: *Mind On Statistics*, 3rd edition, Utts and Heckard).

Lab 2

Read in this DATA:

```
pd.read_csv('https://raw.githubusercontent.com/fenago/pythonml/main/data/lungfunction.txt',sep="\t")
```

Apply what you have learned to create Simple Linear Models.

This dataset of size $n = 51$ are for the 50 states and the District of Columbia in the United States (poverty.txt). The variables are y = year 2002 birth rate per 1000 females 15 to 17 years old and x = poverty rate, which is the percent of the state's population living in households with incomes below the federally defined poverty level. (Data source: *Mind On Statistics*, 3rd edition, Utts and Heckard).

Multivariate Linear Regression

Our Dataset

Data Dictionary:

http://people.sc.fsu.edu/~jburkardt/datasets/regression/x16.txt

Dataset:

pd.read_csv("https://raw.githubusercontent.com/fenago/pythonml/main/data/petrol_consumption.csv")

Load the Libraries

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline
```

Load the Data

```
df =
pd.read_csv("https://raw.githubusercontent.com/fenago/pythonml/main/data/petrol_consumption.csv")

df.head()

df.describe()
```

Find Your Correlations In your Datasets

```
# correlation between 2 Specific Columns
print(df['Petrol_tax'].corr(df['Petrol_Consumption']))



# pair-wise correlation between all columns

print(df.corr())
```

Prepare the Data

```
X = dataset[['Petrol_tax', 'Average_income',
'Paved_Highways','Population_Driver_licence(%)']]

y = dataset['Petrol_Consumption']


#Execute below to divide into train/test sets

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=0)
```

Train the Algorithm as Before

```
from sklearn.linear_model import LinearRegression

regressor = LinearRegression()

regressor.fit(X_train, y_train)
```

What Coefficients Did it Find?

```
coeff_df = pd.DataFrame(regressor.coef_, X.columns,
columns=['Coefficient'])
```

```
coeff_df
```

	Coefficient
Petrol_tax	-24.196784
Average_income	-0.81680
Paved_Highways	-0.000522
Population_Driver_license(%)	1324.675464

Predictions

```
y_pred = regressor.predict(X_test)

df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})

df
```

	Actual	Predicted
36	640	643.176639
22	464	411.950913
20	649	683.712762
38	648	728.049522
18	865	755.473801

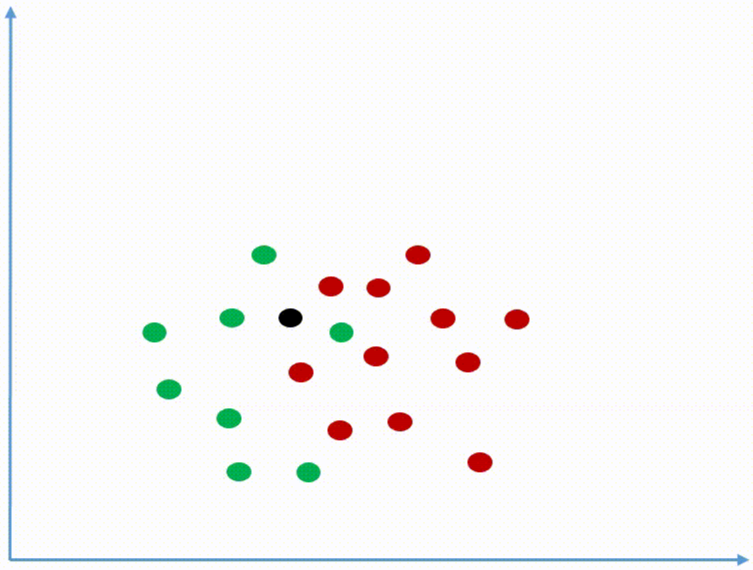
Evaluate the Algorithm

```
from sklearn import metrics

print('Mean Absolute Error:',
metrics.mean_absolute_error(y_test, y_pred))

print('Mean Squared Error:',
metrics.mean_squared_error(y_test, y_pred))

print('Root Mean Squared Error:',
np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

print('R-Squared:',metrics.r2_score(df['Actual'],
df['Predicted']))
```

KNN

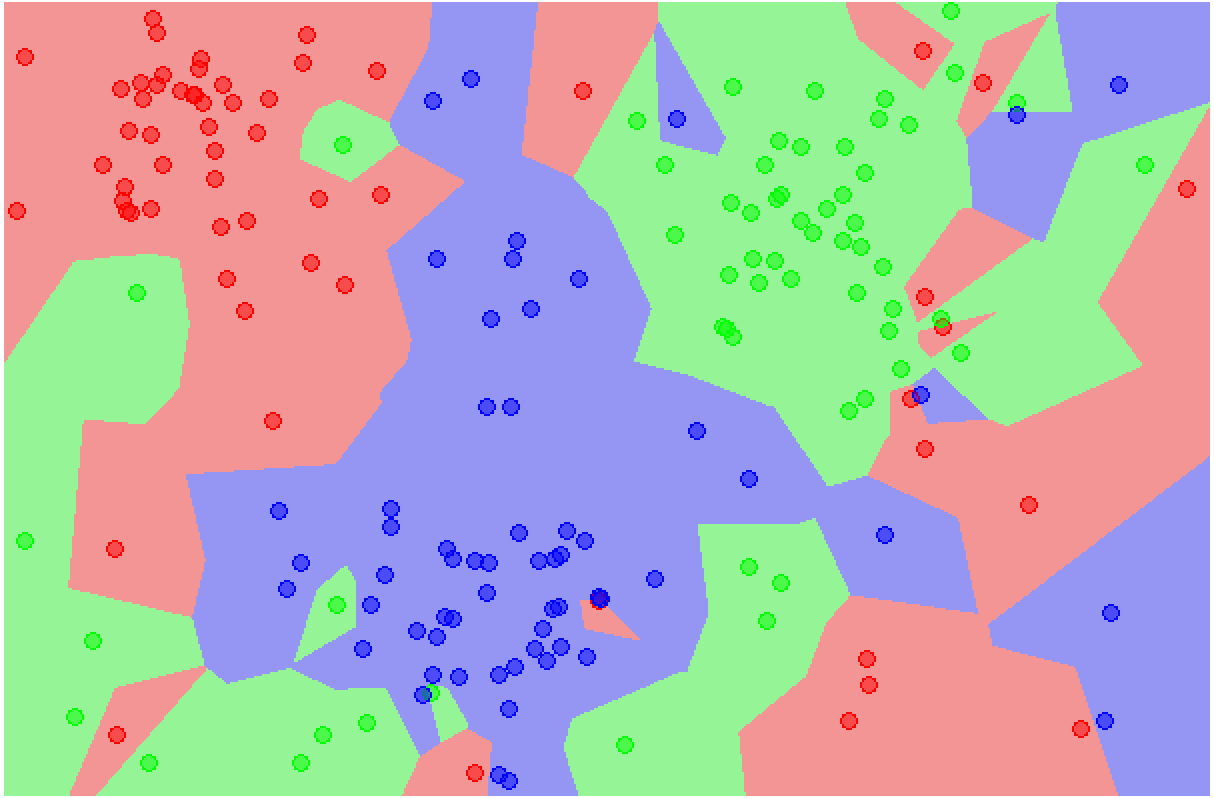
KNN

Choice of value of K



machinelearningknowledge.ai

How Does KNN Work?

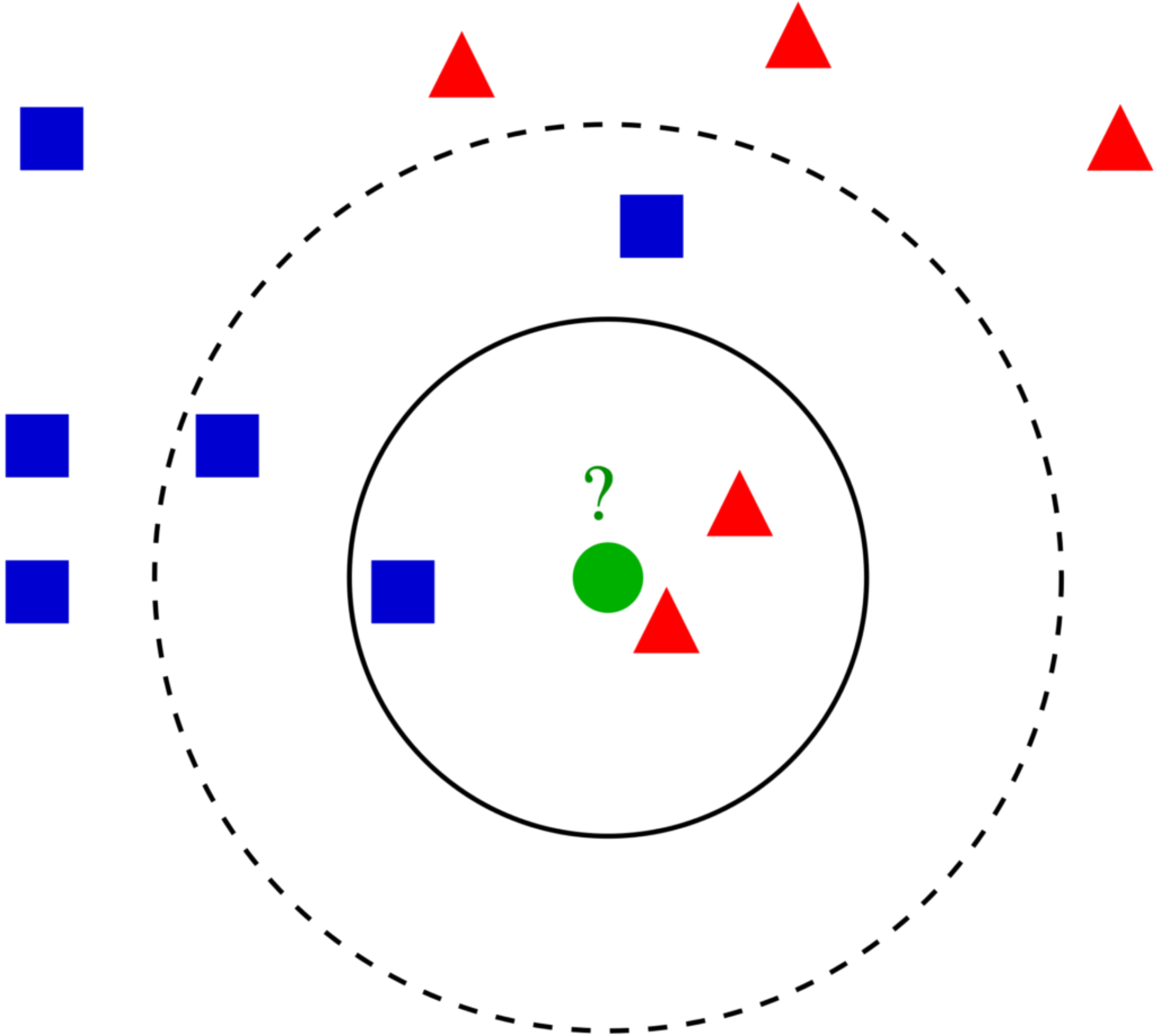


Distance in KNN

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + \cdots + (q_n - p_n)^2}$$

$$= \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}.$$

Nearest?



Steps to Solve a KNN Problem

1. Load and store the data.
2. Calculate the distance from x (new data point) to all other data points.
3. Sort all the distances from your data in ascending order.
4. Initialize the K value for the nearest data points.
5. Make a prediction based on the majority of data points with the same label within the K value.
6. Evaluate your machine learning model.

The Use Case...

we want to create  allow botanists
to classify diffe s.