Multidisciplinary Design Program

Fundamentals of Machine Learning

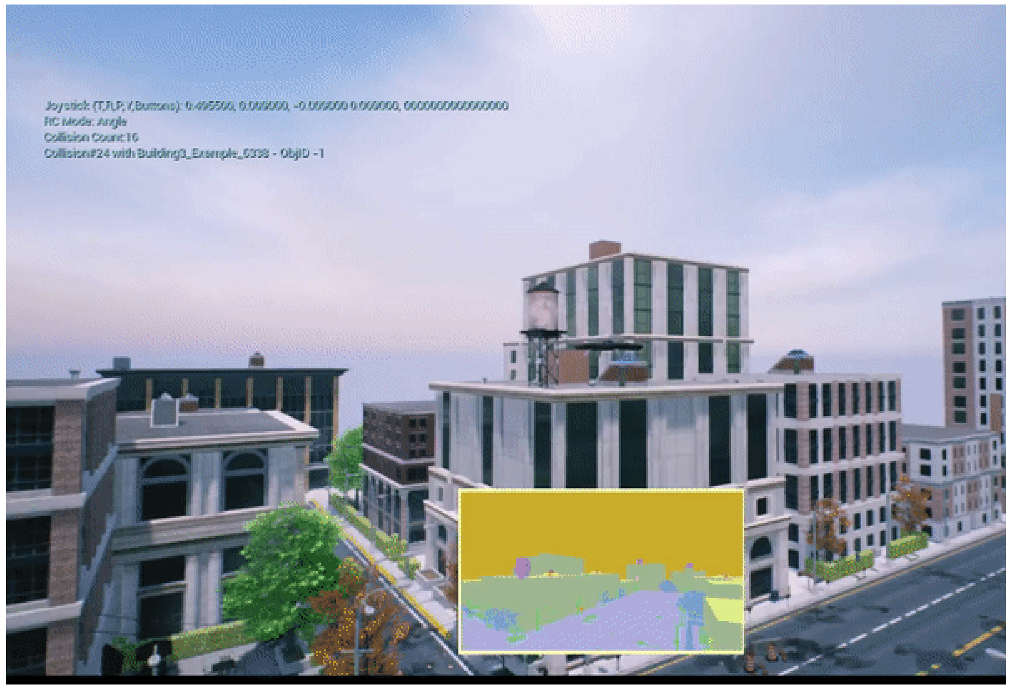
Jeremy Castagno

01/30/2019

About Me - Jeremy Castagno

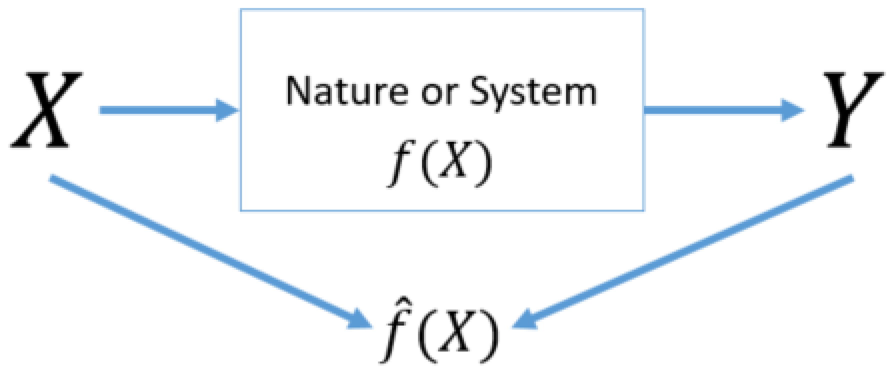
- 2013 - BSc. in Chemical Engineering at BYU
- Valero Energy Corporation as process control engineer
- 2016 – Robotics PhD University of Michigan

- Topic interests - Artificial Intelligence, Machine Learning, Path Planning, Simulation
- Research topic – Unmanned Aerial System (UAS) emergency rooftop landing in urban cities



What is Machine Learning

- Statistical Learning, Pattern Recognition, Big Data, Data Mining, Expert Systems, Artificial Intelligence (AI), Deep Learning
- Definition- Algorithms and Statistical Models to perform a specific task
- Broadly speaking there are three categories of ML- **supervised** learning, **unsupervised** learning, and **reinforcement** learning
- In this *short* class we will focus only on **supervised** and **unsupervised** learning
- In almost all cases there is an input (x) and an output (y) for our system



There is a true $f(x)$ which we seek to approximate with an $\hat{f}\,(x)$

MDP Projects

- Analyzes audio inputs, with a goal of determining factors such as occupants, locations, and state of vehicle
- Process the . . . road itself . . . to identify free paths or drivable surfaces
- Recognize . . . email . . . and direct email to appropriate response functions [classification] as well as suggesting a generated response

Basic Terms

- Regression vs Classification
    - Regression - Output takes continuous valued variables
    - Classification - Output determines group membership (class)
    - Sometimes the output is **both** - Semantic bounding boxes
- Features, predictors, response/independent variables - text email, audio stream, video stream
- Classes, labels, ground truth
- Deep learning and neural networks
    - A cascade of multiple layers processing units
    - Each layer(s) may learn different abstractions of the task

Supervised Learning

Learning a function that maps an *input* (X) to and *output* (Y)

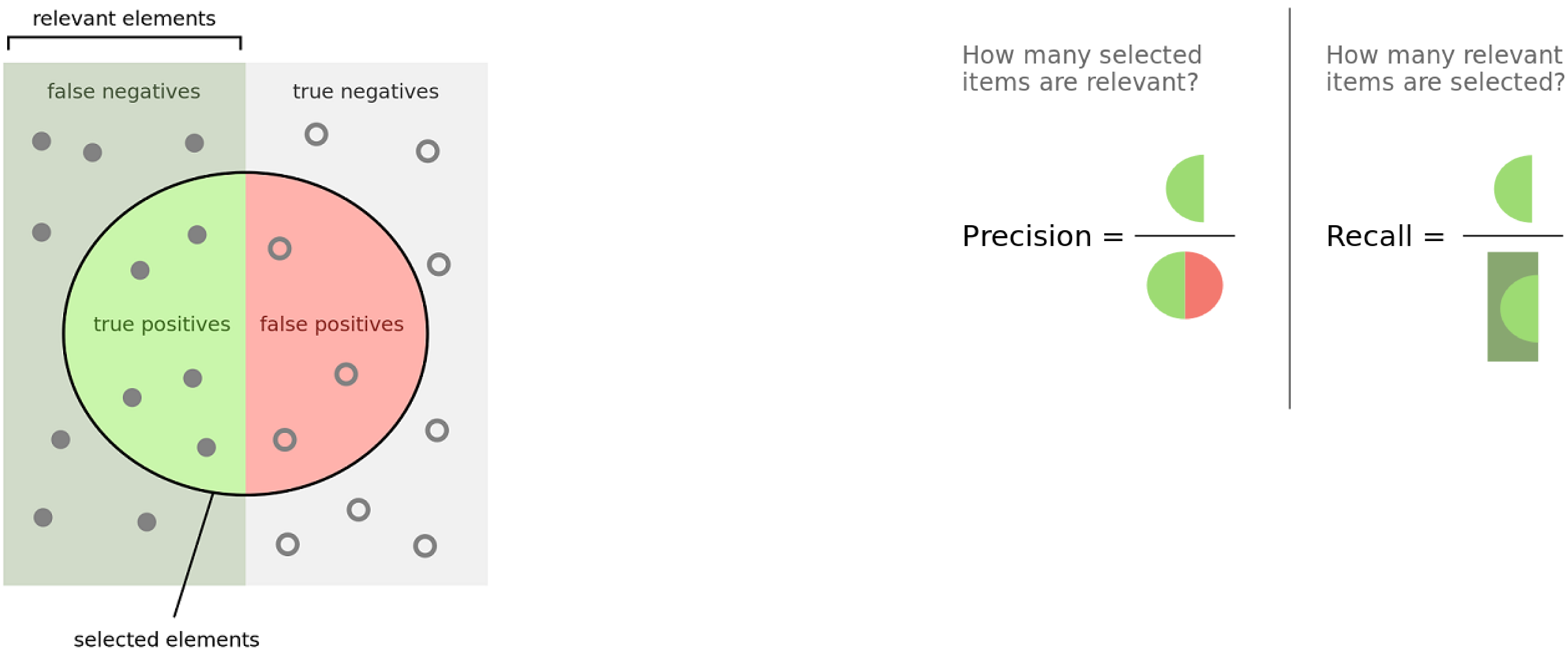
$$Y = f(X) + \underbrace{\epsilon}_{error}$$

$$\mathbf{X} = \overbrace{\begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}}^{Features} \qquad \mathbf{Y} = \overbrace{\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}}^{labels}$$

- Main Questions:
  - Which predictors are associated with the response?
    - Should we preprocess our raw data into features? Text $\Longrightarrow$ Vector
  - What is the relationship between the response and each predictor?
  - What model can be used to estimate f?
    - Parametric Functions vs Non Parametric Functions
    - Only focus on parametric function in this class

Assessing Performance

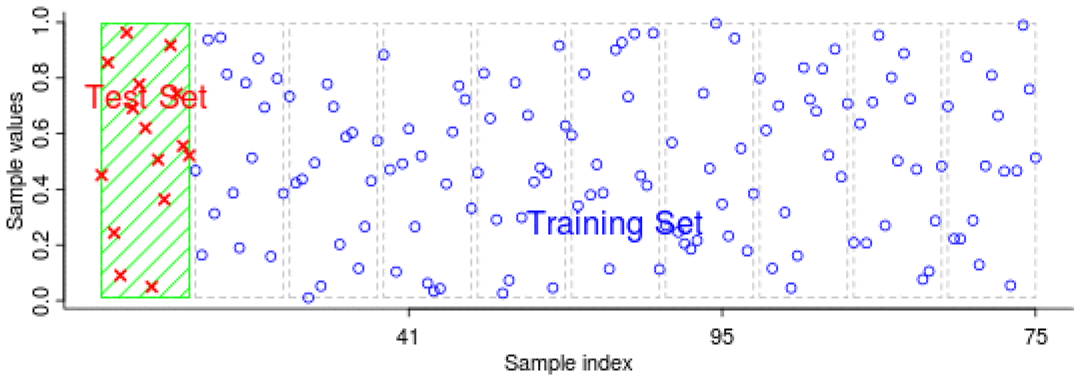
- Training Set and Test Set - Split 60/40
  - Your model should never be trained with the test set
- Regression
  - Mean Squared Error (MSE) = $MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \hat{f}\ (x_i)\right)^2$
- Classification
  - Training use Logarithmic Loss - $\frac{-1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij} * \log(p_{ij})$
  - Final Assessment
    - Classification Accuracy = $\frac{\text{\# of correct pred.}}{\text{Total \# of pred.}}$
    - Confusion Matrix - True Positive, True Negative, False Positives, False Negatives



$$F_1 = \left(\frac{\text{recall}^{-1} + \text{precision}^{-1}}{2}\right)^{-1} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

- Many others for specific tasks
  - Image Bounding Boxes - Intersection of Union (IOU)

- k-Fold cross validation - Train model on subsets of your training data
  - Randomly divide data set into *k* folds of equal size.
  - The first fold is treated as a validation set, the remaining *k* - 1 data is trained on
  - repeat *k* times
  - This results in *k* estimates of validation error
  - $\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i$



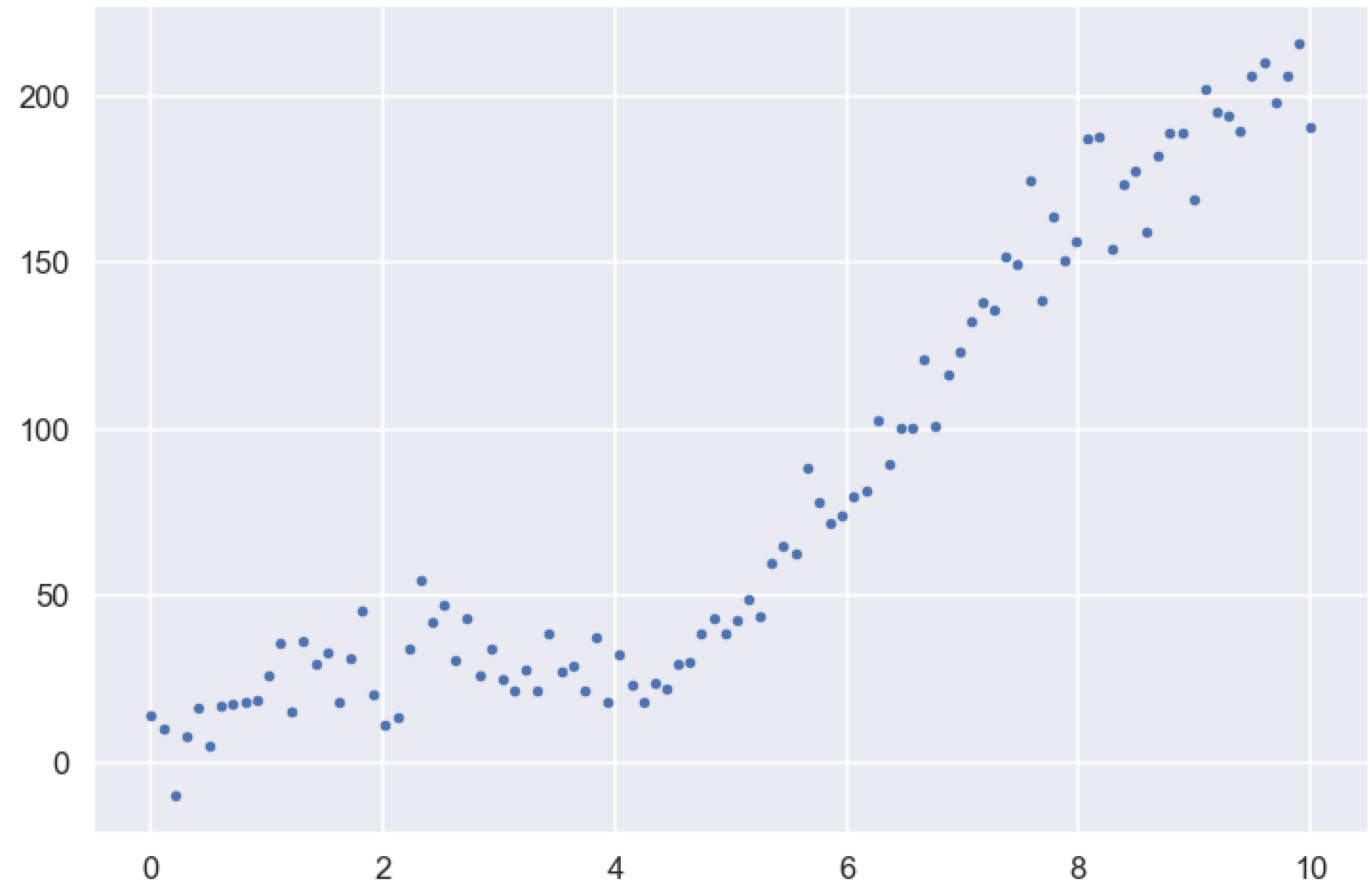
[Source (https://imada.sdu.dk/~marco/Teaching/AY2010-2011/DM825/)](https://imada.sdu.dk/~marco/Teaching/AY2010-2011/DM825/)

Simple Regression Model

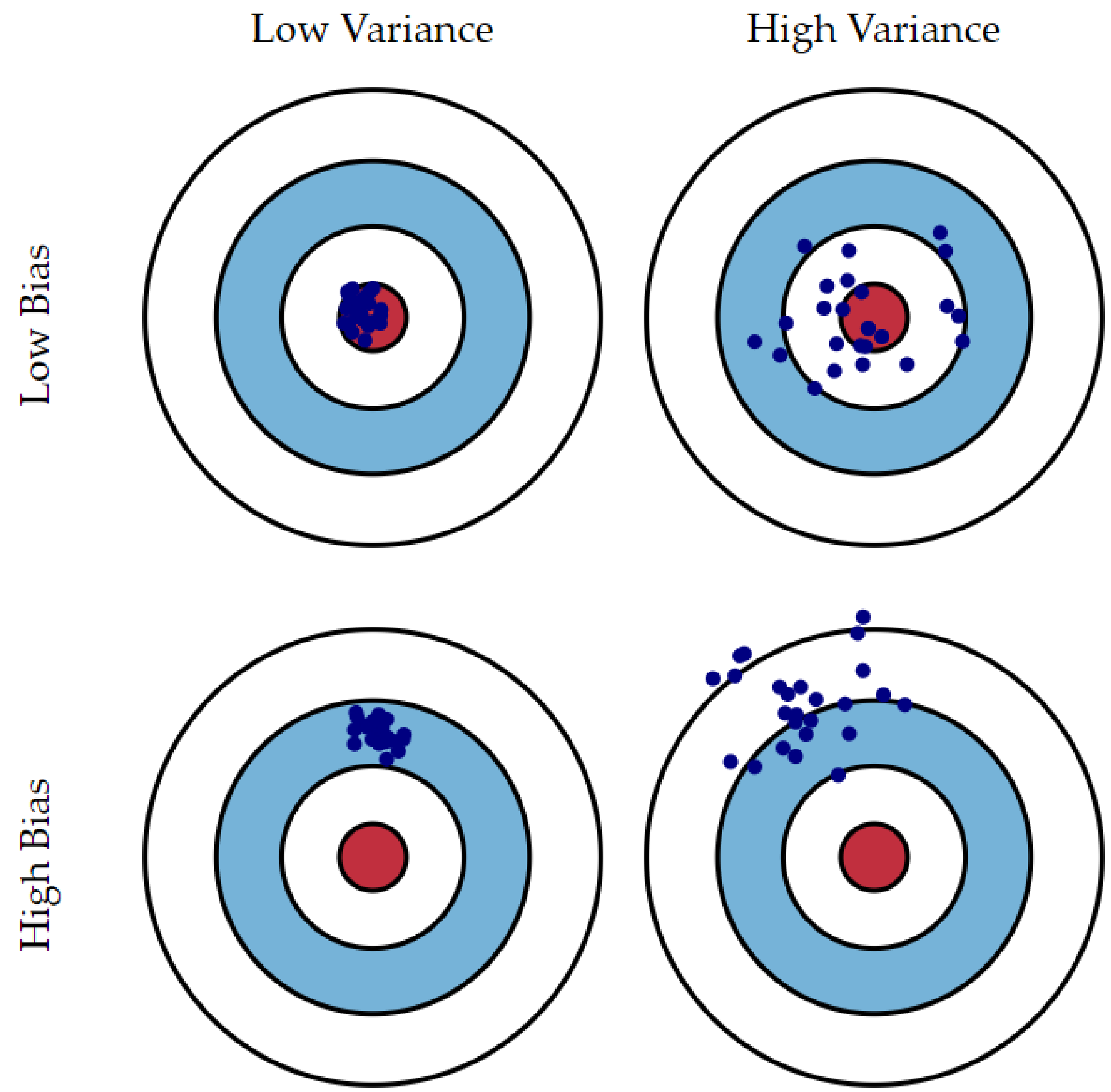
Goals

- Understand bias/variance tradeoff
- Techniques to prevent overfitting

Question: How would you fit this data?



Bias vs Variance



Source: Scott Fortmann (http://scott.fortmann-roe.com/docs/BiasVariance.html)

$$Y = f(X) + \underbrace{\epsilon}_{error}$$

$$\mathrm{Err}(x) = E\left[(Y - \hat{f}\,(x))^2\right]$$

$$\mathrm{Err}(x) = \underbrace{\left(E[\hat{f}\,(x)] - f(x)\right)^2}_{\text{Bias}^2} + \underbrace{E\left[(\hat{f}\,(x) - E[\hat{f}\,(x)])^2\right]}_{\text{Variance}} + \underbrace{\mathrm{Var}(\epsilon)}_{\text{Irreducible Error}}$$

- Bias indicates a fundamental mismatch between the actual function, $f(x)$, and the estimated function $\hat{f}(x)$
  - Linear model trying to estimate a quadratic function
- Variance refers to the amount by which $\hat{f}$ would change if we estimated it using a different training data set
  - Each $\hat{f}$ would be the same fundamental model, but have different parameters associated with it.
  - In a perfect world these different estimates of $\hat{f}$ would vary very little.
  - If a model has high variance, small changes in the training data will make big changes in $\hat{f}$
- $\mathrm{Var}(\epsilon)$ is noise in the data and can never be reduced
- More flexible the model then variance will increase and the bias will decrease

In [35]:

```
interact(plot_interact,  mu=fixed(0), sigma=widgets.IntSlider(min=0,max=30,step=5,value=20),
         num_samples=widgets.IntSlider(min=100,max=500,step=200,value=100));
```

Simple Classification Model

- Iris dataset - https://archive.ics.uci.edu/ml/datasets/Iris (https://archive.ics.uci.edu/ml/datasets/Iris)
- Features - Sepal Length, Sepal Width, Petal Length, Petal Width
- Output - Class prediction (Setosa, Veriscolor, Virginica)

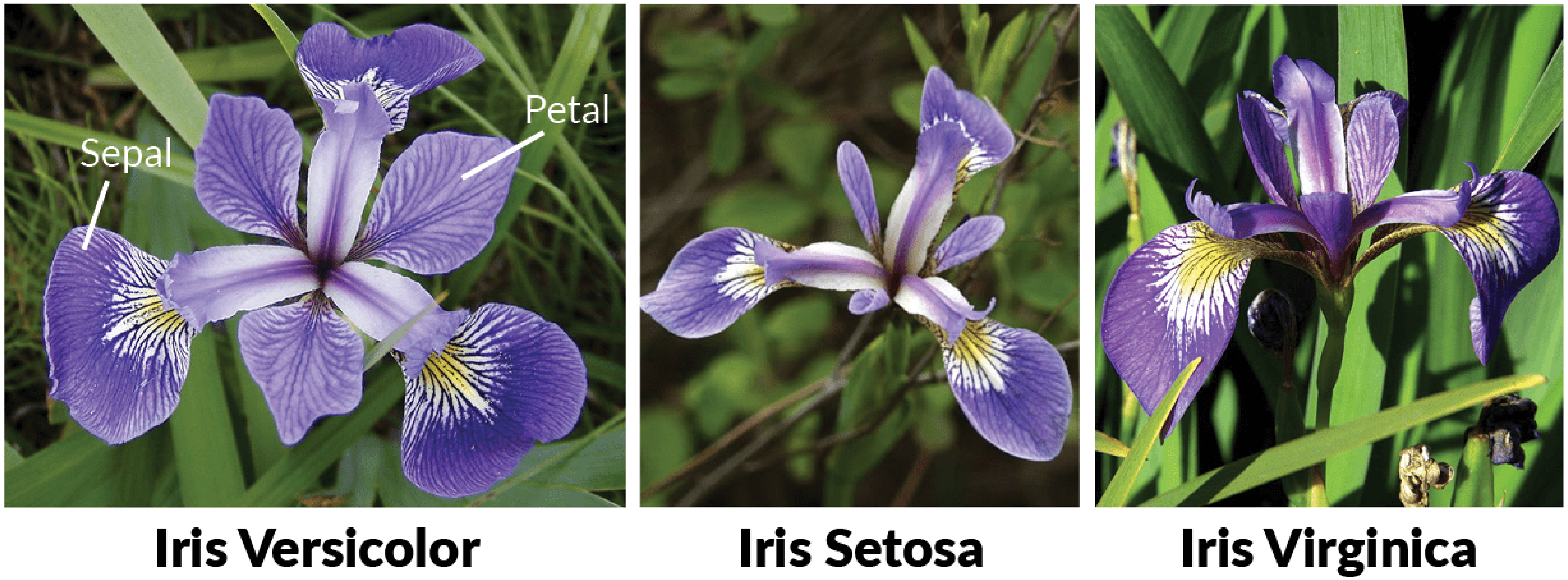


Image - https://www.datacamp.com/community/tutorials/machine-learning-in-r (https://www.datacamp.com/community/tutorials/machine-learning-in-r)

View of the Data

In [5]:
```
df_iris = sns.load_dataset('iris')
display(HTML(df_iris.head().to_html(index=False)))
```

sepal_length	sepal_width	petal_length	petal_width	s
5.1	3.5	1.4	0.2	s
4.9	3.0	1.4	0.2	s
4.7	3.2	1.3	0.2	s
4.6	3.1	1.5	0.2	s
5.0	3.6	1.4	0.2	s

In [6]:
```
fig, ax = plt.subplots(figsize=(10,8), nrows=1, ncols=1)
sns.swarmplot(x="measurement", y="value", hue="species",
              palette=["r", "c", "y"], data=iris, ax=ax)
```

Out[6]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x27759579e80>
```

What are common techniques for separating classes?

- Linear Regression? No!

- Predict a number between 0-2

Species	**Encoding**
Setosa	0
Veriscolor	1
Virginica	2

- Whats wrong with this?

- It creates an **ordering** to the classes. The regression learns that *Virginica* is "closer" to *Verisicolor* than *Setosa*.
- Often this ordering is **not** what we want

- One Hot Encoding to the rescue!
- Turn C classes into an **array** of size C

y	**Sertosa**	**Versicolor**	**Virginica**
Label 1	1	0	0
Label 2	0	1	0
Label 3	0	0	1

- Our model should learn to match this vector
- Model output is a probability distribution $\begin{pmatrix} .90 & .05 & .05 \end{pmatrix}$
- If you only have two classes (binary) use one number to represent probability of both classes.

Techniques we will learn

- Logistic Regression
- Support Vector Machines
- Random Forest

Techniques you should research later

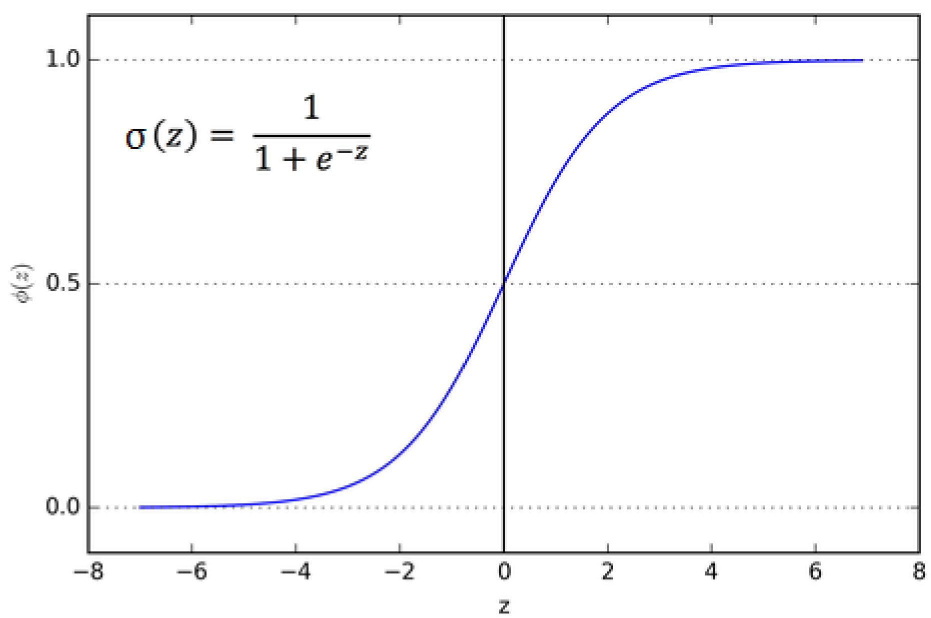
- XGBoost

Logistic Regression

Logistic Regression - Binary

Binary Case (0 or 1)

- x is our feature vector: [Sepal Len, etc..]
- w is a parameter vector. $w_0 x_0 + w_1 x_1 + ... + b$
- $\sigma(z) = \frac{1}{1+e^{-z}}$



$$z = w^T x + b$$
$$p(y=1) = \sigma(z) = \sigma(z)$$
$$z = \log \frac{p(y=1)}{1-p(y=1)}$$

$$\begin{aligned} \text{loss} &= -y\log(\sigma(z)) - (1-y)\log(1-\sigma(z)) \\ &= \begin{cases} -\log(1-\sigma(z)) & y=0 \\ -\log(\sigma(z)) & y=1 \end{cases} \end{aligned}$$

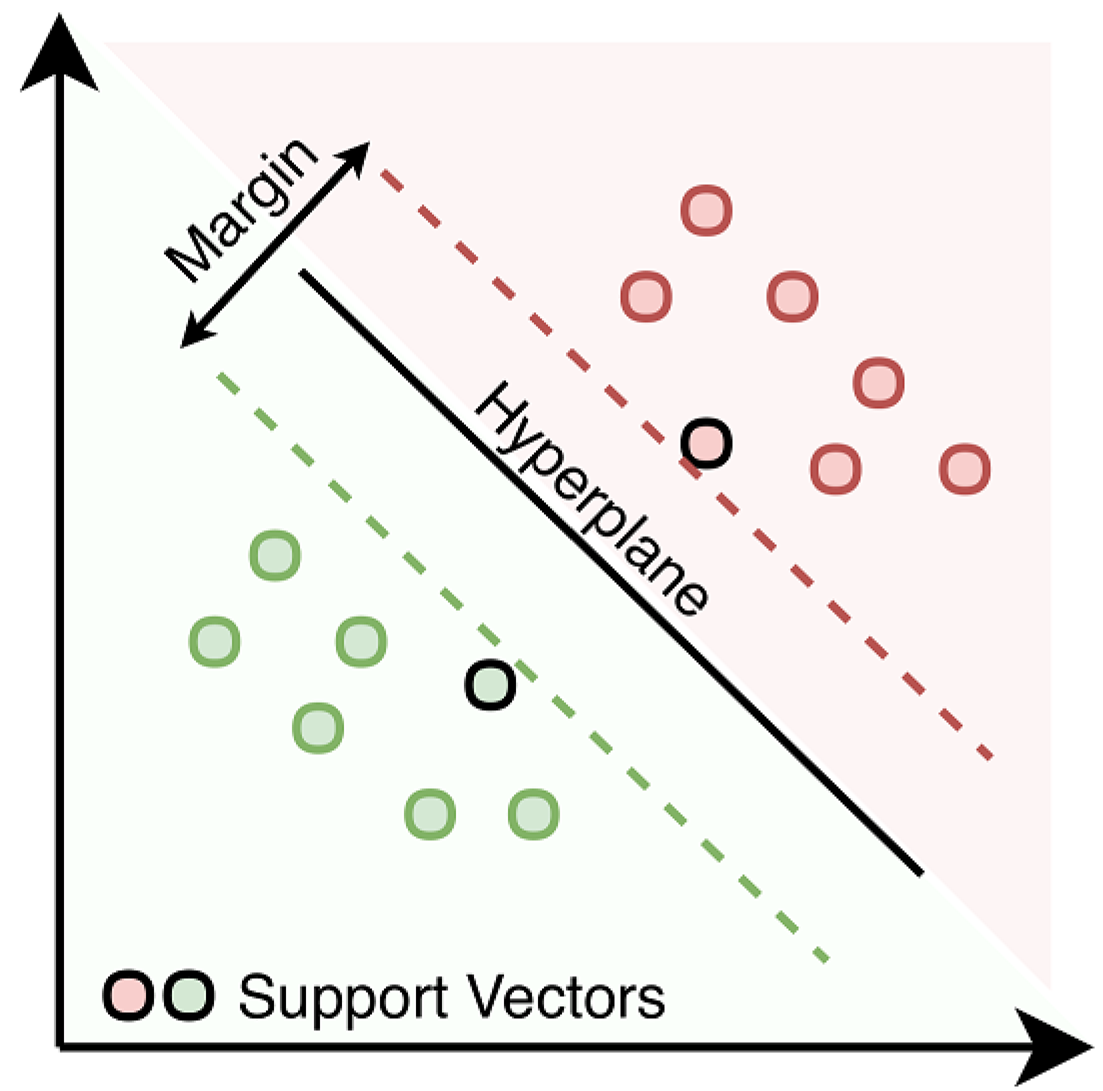
Minimize the loss by manipulating *w*. Use a solver!

Logistic Regression - Multiple Classes

- Assume we have C classes
- One vs all
  - Separately train C *binary* classifiers
  - Select the one with the highest probability
- Multinomial Logistic Regression
  - Use one hot encoding.
  - Train all classifiers together, minimizing their combined loss

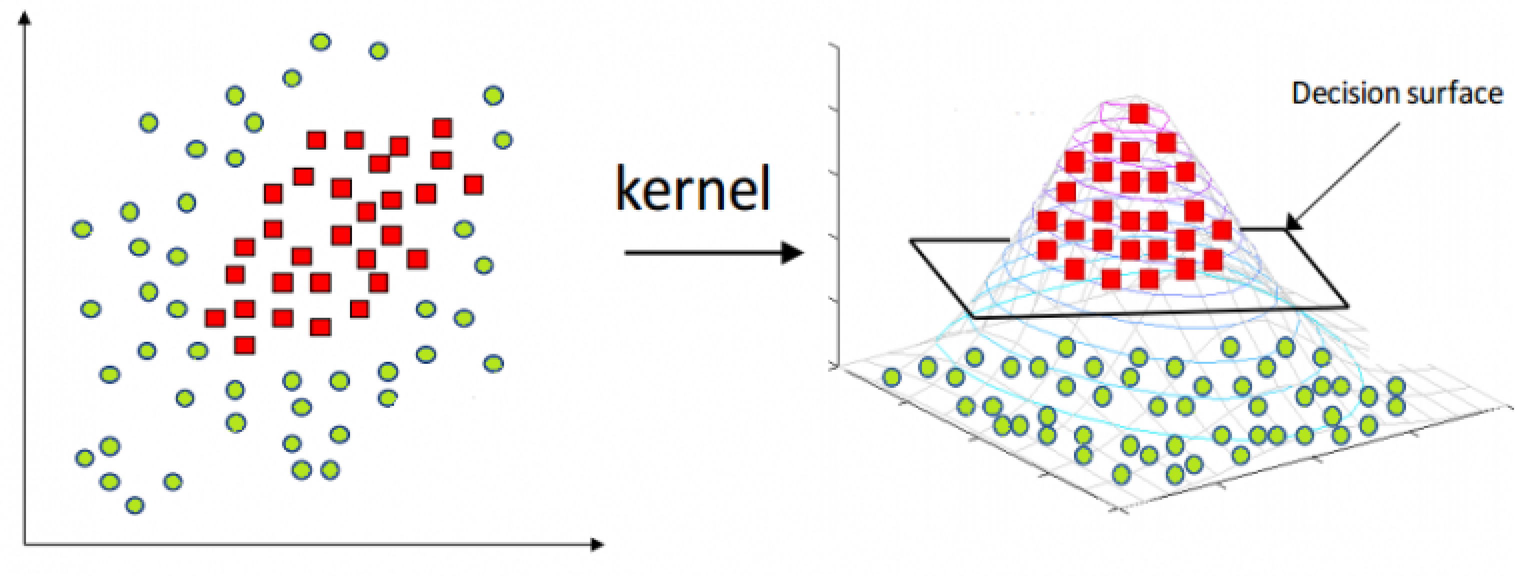
Support Vector Machines

- SVM's separate binary classes through a linear separating hyperplane
- The plane is constructed by solving a global optimization problem
- Maximize the margin M while minimizing misclassified values
- Most data can **not** be separated *completely*, so a tuning parameter, C is used to specify the *slack*



Kernel Trick

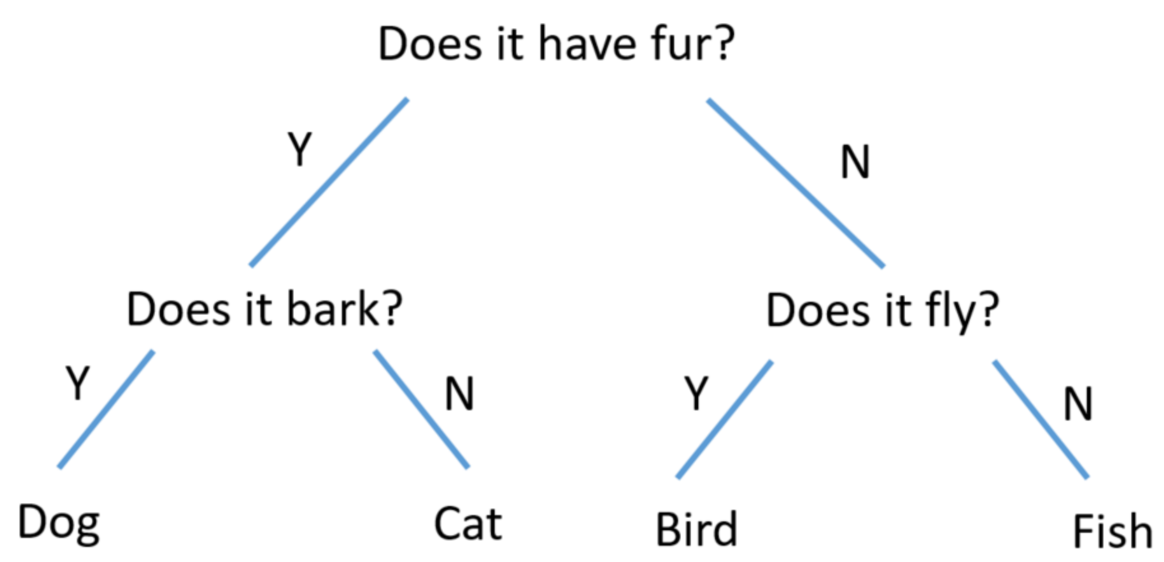
- Separation is done by computing dot products between features
- Some data is not linearly separable from the provided features
- Map your feature spaces to higher dimensions, $\Phi(X)$
- $K(P(x1, y1), P(x2, y2)) = x_1^2 x_2^2 + y_1^2 y_2^2 + 2x_1 y_1 * x_2 y_2$
- $x_1^2 x_2^2 + y_1^2 y_2^2 + 2x_1 y_1 * x_2 y_2 =< \left(x_1^2, y_1^2, \sqrt{2x_1 y_1}\right), \left(x_2^2, y_2^2, \sqrt{2x_2 y_2}\right) >$



HackerEarth (https://www.hackerearth.com/blog/machine-learning/simple-tutorial-svm-parameter-tuning-python-r/)

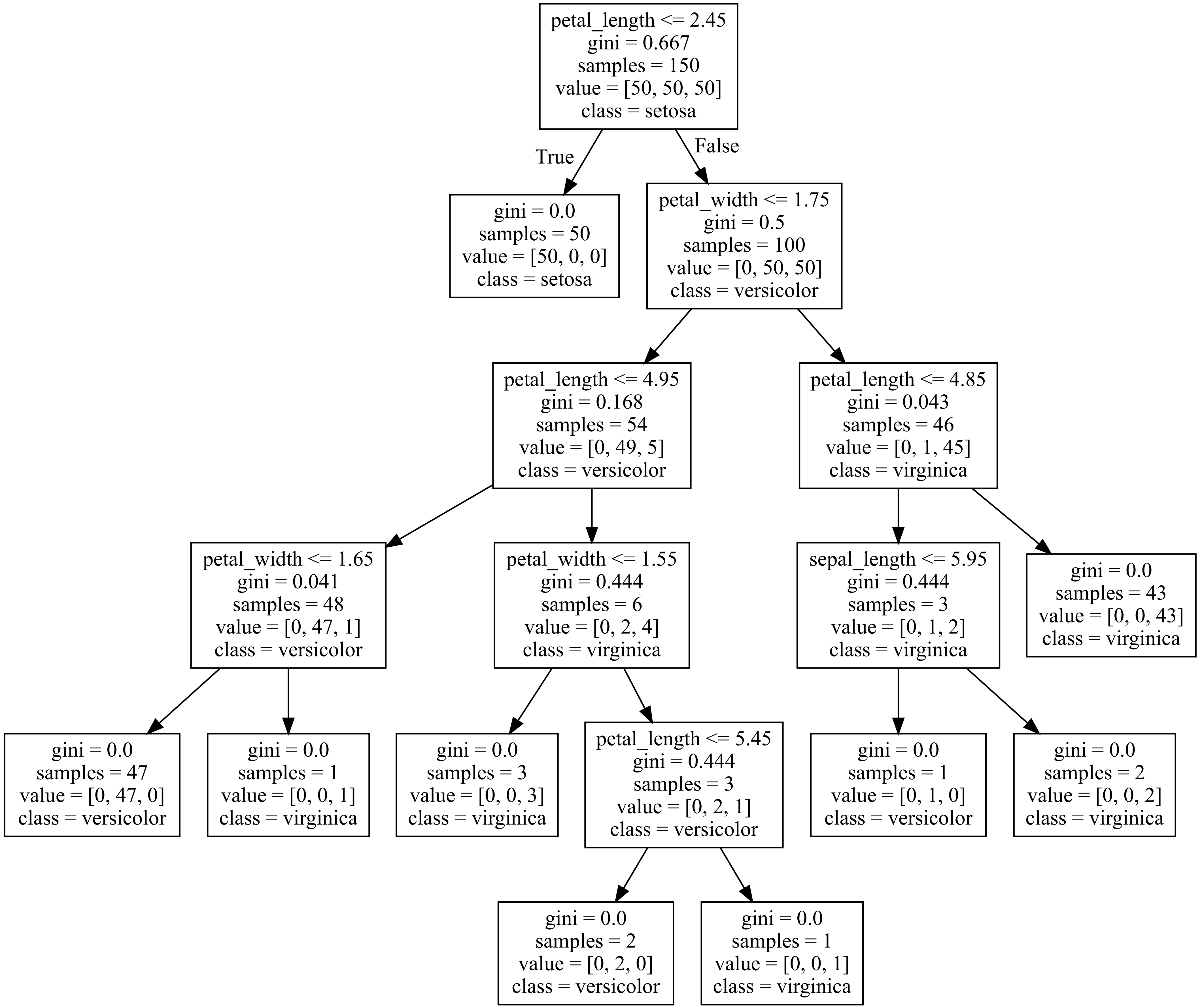
Decision Trees

- A decision tree is drawn upside down with its root at the top.
- Leaves represent class labels and branches represent conjunctions of features that predict class labels
- Construction - choose a feature at each step that best splits the set of items
  - Entropy
  - Gini
- Repeat this process until all training data is split



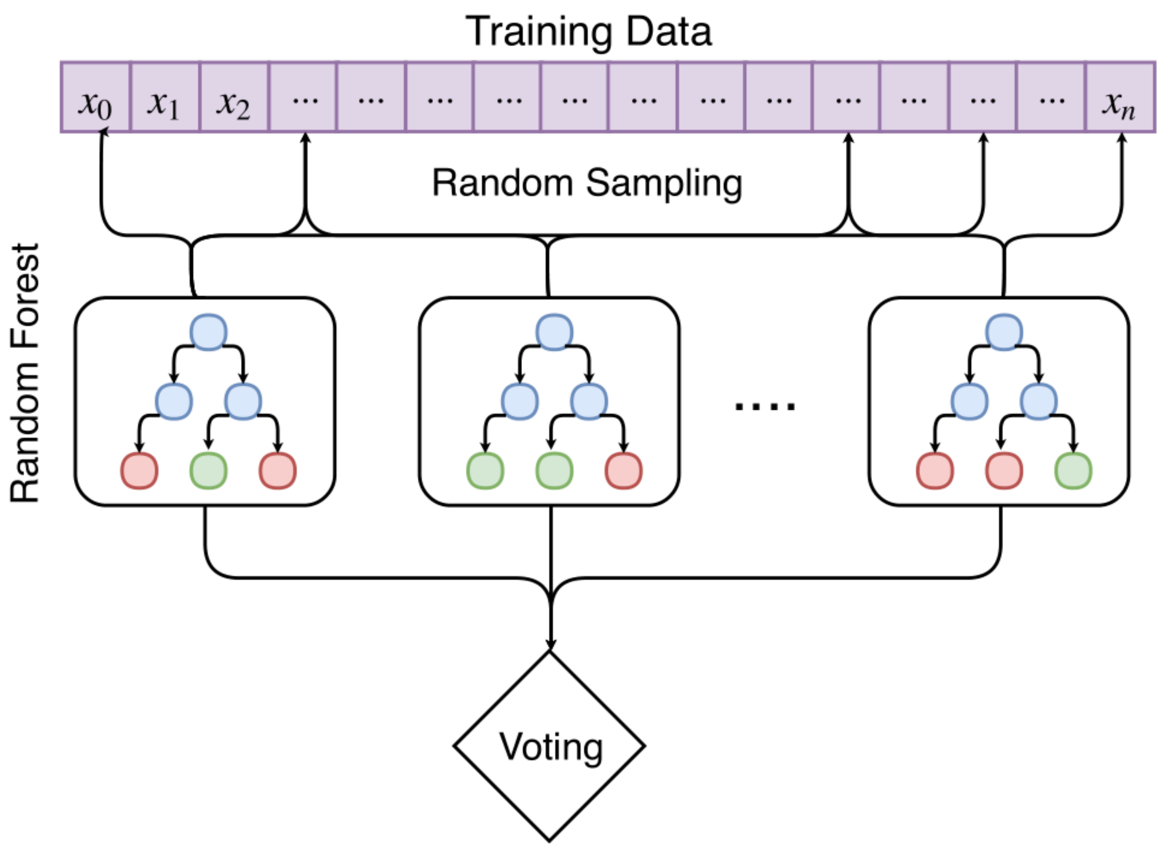
In [8]:
```
graph
```

Out[8]:



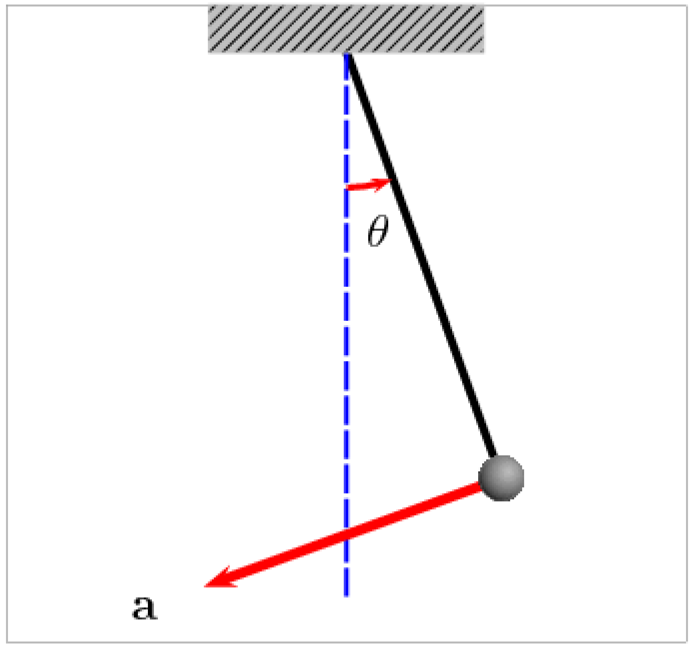
Random Forests

- Ensemble Learning - A collection of multiple independently trained learning algorithms
- Bagging - Random sampling of training data
- Random feature selection (if there are alot of features)



Break

- Restroom
- Discuss Projects



Natural Language Processing

What makes NLP hard

- Raw data is not in a vector space! Characters are not numbers and don't directly relate to eachother.
    - A lot of great work in transforming text into vector spaces using Deep learning.
- Computer can count patterns and such, but don't innately understand concepts that words and phrases provide.
    - There is a conceptual dimension that must be learned.
    - It's difficult to construct and train a model that does this with language.
- Incredible amount of ambiguity in text, even for a human.

Techniques we will learn

- Basic Fundamental Steps of NLP
    - Example - Spam Analysis from SMS Text Messages
- Deeper Feature Extraction - Spacey
    - Word Vectors
    - Part-of-speech tags, Named entity labels

Basic Fundamentals of NLP

1. Look at data
2. Preprocess Text
3. Tokenization
4. Compute TF-IDF scores
5. Choose and Fit Models
6. Hyperparameter Search with Cross Validation

References: In Machines We Trust (http://inmachineswetrust.com/posts/sms-spam-filter/), SKLearn (https://scikit-learn.org/stable/modules/cross_validation.html) Machine Learning Mastery (https://machinelearningmastery.com/clean-text-machine-learning-python/)

Data set comes from UCI and is a public set of SMS labeled messages

Data breakdown:

Ham	Spam
4825	747

https://archive.ics.uci.edu/ml/datasets/sms+spam+collection (https://archive.ics.uci.edu/ml/datasets/sms+spam+collection)

In [10]:
```
df_spam.head()
```

Out[10]:

	Label	Text
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Look at Data

In [12]:
```
display(HTML(df_examples.to_html(index=False)))
```

Label	Text
ham	Awesome, I'll see you in a bit
ham	Hey j! r u feeling any better, hopeSo hunny. i amnow feelin ill & ithink i may have tonsolitusaswell! damn iam layin in bedreal bored. lotsof luv me xxxx
spam	You are guaranteed the latest Nokia Phone, a 40GB iPod MP3 player or a £500 prize! Txt word: COLLECT to No: 83355! IBHltd LdnW15H 150p/Mtmsgrcvd18+
spam	XMAS iscoming & ur awarded either £500 CD gift vouchers & free entry 2 r £100 weekly draw txt MUSIC to 87066 TnC www.Ldew.com1win150ppmx3age16subscription

Preprocess Text

- Encode the ham/spam label as 0 or 1.
- Remove Punctuation
- Leading and Ending White space - ' Hey '
- Replace common occurring text patterns with a single word - Regular Expression
  - 'http://spam.me (http://spam.me)' --> url
  - '$' '£'& --> 'mnsymb'
  - '55555' -- 'shrtcode'
  - '867-5309' --> 'phonenumber'
  - '88' --> 'number'
- Lower case
- Port Stemmer - 'testing' -> 'test'
- Stop words - 'the', 'a'

In [15]:
```
preprocess_text('TEXT me at 801-458-3434 to win prize at www.spam.com')
```
Out[15]:
```
'text phonenumb win prize url'
```
In [16]:
```
display(HTML(df_spam.iloc[examples].to_html(index=False)))
```

Label	Text	clean_text	label_encoded
ham	Awesome, I'll see you in a bit	awesom I see bit	0
ham	Hey j! r u feeling any better, hopeSo hunny. i amnow feelin ill & ithink i may have tonsolitusaswell! damn iam layin in bedreal bored. lotsof luv me xxxx	hey j r u feel better hopeso hunni amnow feelin ill ithink may tonsolitusaswel damn iam layin bedreal bore lotsof luv xxxx	0
spam	You are guaranteed the latest Nokia Phone, a 40GB iPod MP3 player or a £500 prize! Txt word: COLLECT to No: 83355! IBHltd LdnW15H 150p/Mtmsgrcvd18+	you guarante latest nokia phone number GB ipod MP number player mnsymb number prize txt word collect No shrtcode ibhltd ldnw number H number p mtmsgrcvd number	1
spam	XMAS iscoming & ur awarded either £500 CD gift vouchers & free entry 2 r £100 weekly draw txt MUSIC to 87066 TnC www.Ldew.com1win150ppmx3age16subscription	xma iscom ur award either mnsymb number CD gift voucher free entri number r mnsymb number weekli draw txt music shrtcode tnc url	1

Tokenization

- Our features will be every term (word) of the corpus of all terms in all documents (examples)
- This assumes that each word is linearly independent of another.
- Are you hungry? vs You are hungry!
- Use n-grams
- Are you hungry => 'are', 'you', 'hungry', 'are you', 'you hungry'

Compute Term Frequency-Inverse document Frequency (TF-IDF)

- TF-IDF is a weight used to evaluate how important a term is to a document (one example) in the collection (all examples)
- Term Frequency, tf
    - Count of how many times a term occurred in the document
- Inverse Document Frequency
    - Measures how important a term is

$$\mathrm{tf}(t, d) = |t \in d|$$

$$\mathrm{idf}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

$$\mathrm{tfidf}(t, d, D) = \mathrm{tf}(t, d) \cdot \mathrm{idf}(t, D)$$

N is count all documents. $|\{d \in D : t \in d\}|$ is the number of documents where the term t appears

In [18]:
```
X_data
```

Out[18]:
```
<5572x39201 sparse matrix of type '<class 'numpy.float64'>'
	with 102112 stored elements in Compressed Sparse Row format>
```

$$5572 \cdot 39201 = 218,427,972$$

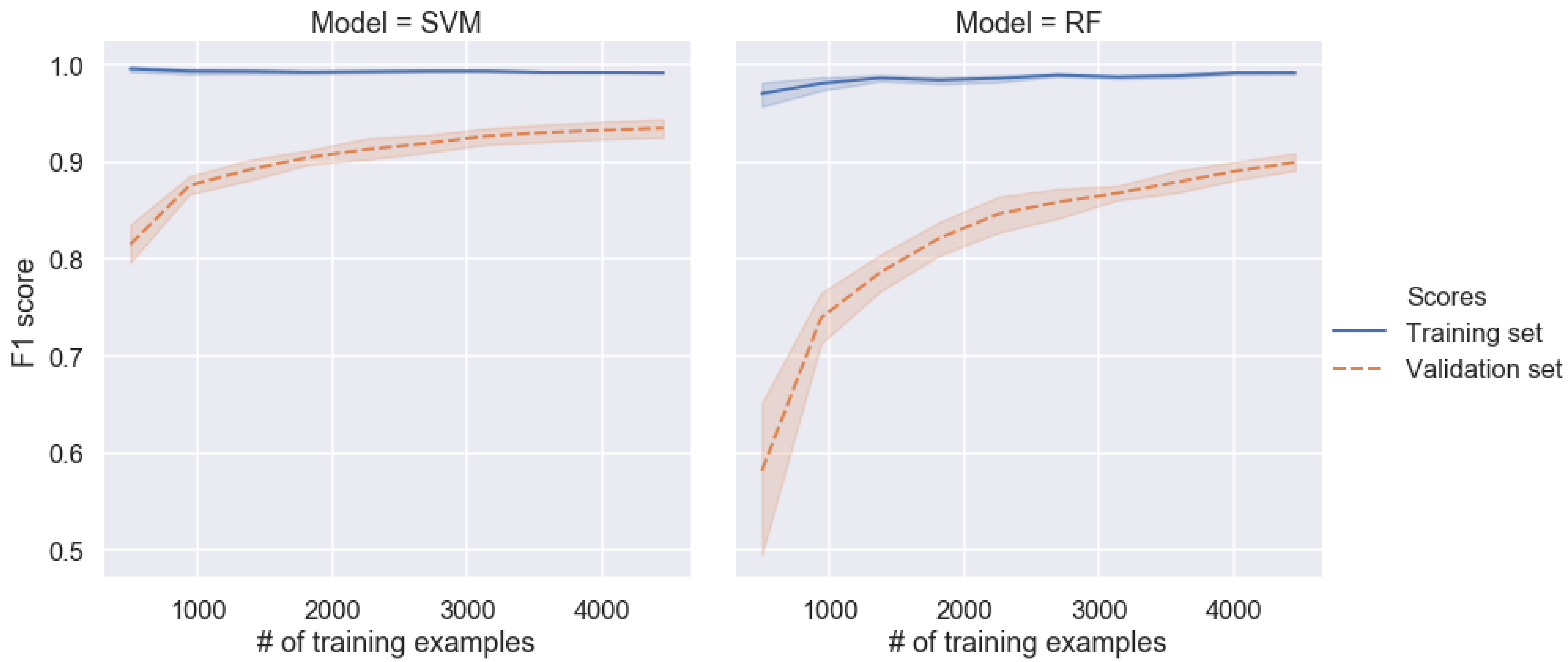
$$\frac{102112}{218{,}427{,}972} = \%0.04$$

Choose Models and fit

- Support Vector Machines
- Random Forest

- Each of these models have many different *hyperparameters* to tune
- Start with defaults parameters for models
  - SVM - Kernel=linear, C=1.0
  - Random Forest - Split-gini, number of estimators = 20

In [22]:
```
g = sns.relplot(x="# of training examples", y="F1 score",
                hue="Scores", style="Scores", col="Model",
                height=6, aspect=1, kind="line", data=df_compare)
```



Hyperparamter Tuning

- There are many hyperparameter to models that can influence their performance
- SVM is influenced by:
  - C - $1e-6, 0.1, 1, 10, 100$
  - Kernel - linear, radial basis function, poly
- We desire to train and validate all **30** combinations of these model parameters
- Choose model with best performance!
- If using python use function called `GridSearchCV` in `sklearn`.

In [25]:
```
print("Optimal Parameters {}".format(clf.best_params_))
print("Test Score: {:.1f}%".format(test_score * 100))
```

```
Optimal Parameters {'C': 10, 'kernel': 'linear'}
Test Score: 92.6%
```

In [26]:
```
report
```

Out[26]:

	f1-score	precision	recall	support
Ham	0.989214	0.981651	0.996894	966.0
Spam	0.925795	0.977612	0.879195	149.0
micro avg	0.981166	0.981166	0.981166	1115.0
macro avg	0.957505	0.979632	0.938045	1115.0
weighted avg	0.980739	0.981112	0.981166	1115.0

Things to improve

- Remove useless words from model
- Spelling correction?
- Specific TF-IDF implementation didn't normalize against document length
- Better feature extraction

NLP Advanced Feature Extraction

- Warning - This next part is farther away from my expertise
- There are many state of the art text feature extractors
- spaCy (https://spacy.io/) - Extremely fast python library
    - Prebuilt models to for large-scale information extraction
    - What is the text about? What is the context? Who, What? When?
    - Textacy (https://chartbeat-labs.github.io/textacy/getting_started/quickstart.html#working-with-text) - A higher level library that wraps around spaCy
- AllenNLP (https://allennlp.org/) - GPU accelerated model training
    - Design and evaluate new deep learning models very quickly
- NLTK (https://www.nltk.org/) - Natural Language Tool Kit
    - "Text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries"

Focus on spaCy and some of its capabilities

- Named Entity Recognition
- Similarity

In [27]:

```python
import spacy

nlp = spacy.load('en_core_web_md')
doc = nlp(u'Hi is this JPMC based in London? I need to report credit card fraud')

for ent in doc.ents:
    print(ent.text, ent.label_)
```

```
JPMC PERSON
London GPE
```

In [29]:

```
tokens = nlp(u'dog cat apple orange')
corr = np.zeros((len(tokens), len(tokens)))
for i, token1 in enumerate(tokens):
    for j, token2 in enumerate(tokens):
        sim = token1.similarity(token2)
        corr[i,j] = sim
df = pd.DataFrame(corr, columns=list(map(str,list(tokens.__iter__()))), index=list(map(str,list(tokens.__iter__()))))
# Generate a mask for the upper triangle
mask = np.zeros_like(df, dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
cmap = sns.color_palette("Blues")

dog_vector = list(tokens.__iter__())[0].vector
```

- You can also convert words into vectors
- Here is an example of the "dog" vector

In [30]:
```
print("Vector Size: {}".format(dog_vector.shape[0]))
print("First 50 elements: \n ", dog_vector[:50])
```

```
Vector Size: 300
First 50 elements:
  [-0.40176   0.37057   0.021281 -0.34125   0.049538  0.2944   -0.17376
 -0.27982   0.067622  2.1693   -0.62691   0.29106  -0.6727    0.23319
 -0.34264   0.18311   0.50226   1.0689    0.14698  -0.4523   -0.41827
 -0.15967   0.26748  -0.48867   0.36462  -0.043403 -0.24474  -0.41752
  0.089088 -0.25552  -0.55695   0.12243  -0.083526  0.55095   0.3641
  0.15361   0.55738  -0.90702  -0.049098  0.3858    0.38      0.14425
 -0.27221  -0.37016  -0.12904  -0.15085  -0.38076   0.049583  0.12755
 -0.082788]
```

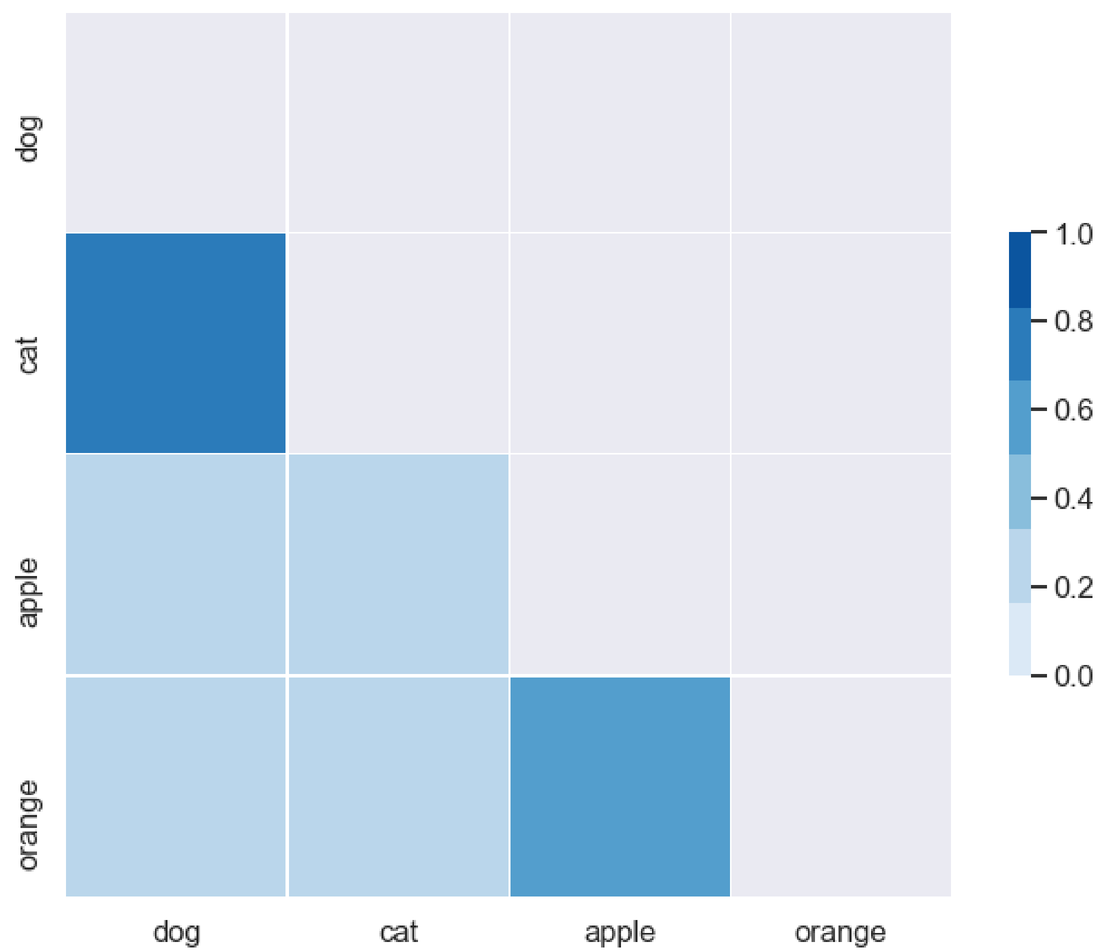
- Here is an example of comparing "dog', "cat", "apple", "orange"

In [31]:

```
# Draw the heatmap with the mask and correct aspect ratio
f, ax = plt.subplots(figsize=(11, 9))
sns.heatmap(df, mask=mask, cmap=cmap,vmin=0, vmax=1.0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5
})
```

Out[31]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x277018d1c18>
```



In [32]:

```
df
```

Out[32]:

	dog	cat	apple	orange
dog	1.000000	0.801685	0.263390	0.274251
cat	0.801685	1.000000	0.282138	0.328847
apple	0.263390	0.282138	1.000000	0.561892
orange	0.274251	0.328847	0.561892	1.000000

Computer Vision