# ML Club Meeting #4

Practice with scikit-learn and decision trees

# For Review please consult your sheet, we have a lot to get through today so please look to your notes for help

# Quick Review Game (use notes)

- The thing we’re predicting -
- Utilizes labeled examples to predict unseen data -
- Predicts continuous values (not discrete) -
- How “bad” the model’s prediction was on a single example -
- How we get to the loss minimization on the loss v. weight graph -

# Scikit-Learn

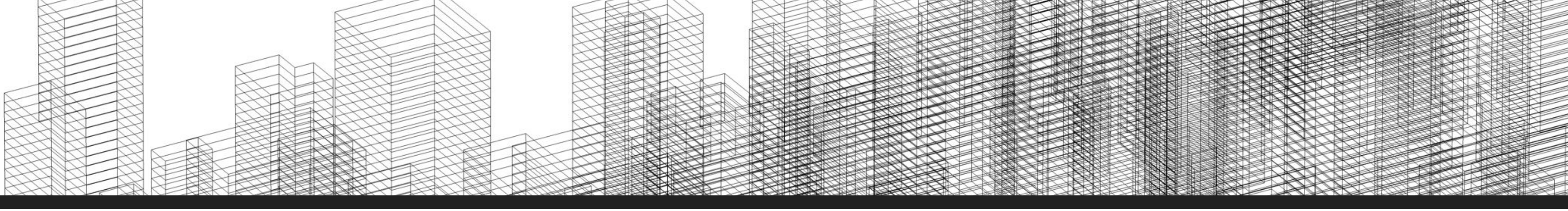
- Popular ML library for basic Machine Learning
- Tool for Data Analysis and Data mining
- Comprised of NumPy and SciPy
    - NumPy - fundamental package for large, multi-dimensional arrays and matrices
    - SciPy - Python library used for scientific computing and technical computing

# Data Sets

- Training set v. Testing set
  - Training set - we learn some properties
  - Testing set - we test the learned properties
- The Scikit package comes with a few standard data sets for both classification and regression

# Quick Installation of Libraries

- For the first half of the meeting we will engage in practice problems
- In the second half, we will introduce ML pipelines with scikit-learn
- Hopefully all of you should have installed Anaconda
- If you do not have a MacOS machine, please share with someone who does …
- Please open your terminal

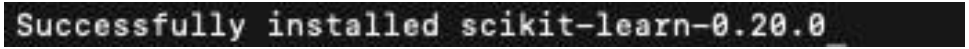


# Scikit-Learn Installation

Review of fruit problem

# Installation of Scikit-learn

- Open terminal and input the following command to update your pip install
  - "`pip install --upgrade pip`"
- After your pip updates or is installed, now input the following line of input
  - "`python -m pip install --user numpy scipy matplotlib ipython jupyter pandas sympy nose`"
- Now input this line of input into your terminal
  - "`pip install -U scikit-learn`"
  - After the system runs this command, you should get a final output like this:

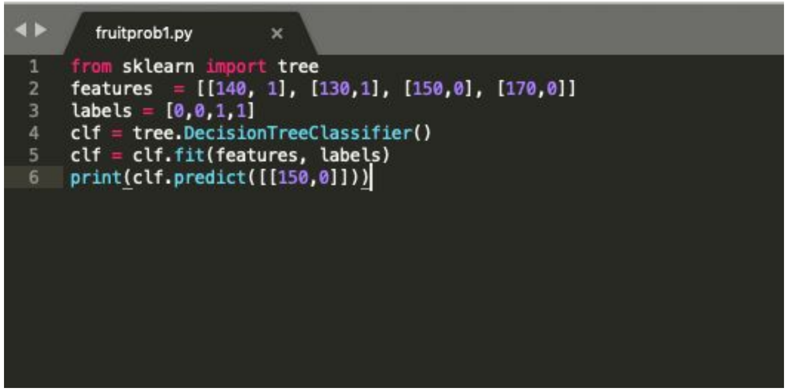


```
Successfully installed scikit-learn-0.20.0
```

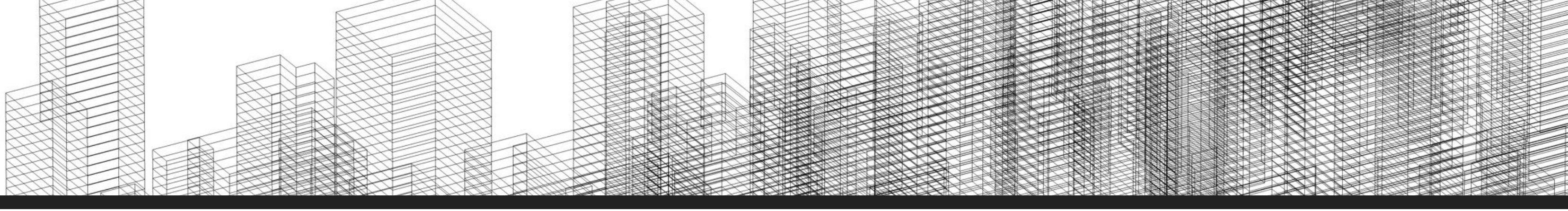
- If that doesn't work, you can try "`conda install scikit-learn`"
  - Please see me if you are having issues (remember - Terminal is case-sensitive)

# Installing Sublime Text

- https://www.sublimetext.com/3
- Make sure you download sublime text for OS X
- As soon as you are done, open a python script in Sublime Text
- Copy the code and save it to desktop as “fruitprob1.py”
  - Don’t forget the “.py” as it is how the system will recognize it as a python file
- Open terminal and input “`cd desktop`”
- Then input “`python fruitprob1.py`”
- You should get an output of “`[1]`”



```python
from sklearn import tree
features  = [[140, 1], [130,1], [150,0], [170,0]]
labels = [0,0,1,1]
clf = tree.DecisionTreeClassifier()
clf = clf.fit(features, labels)
print(clf.predict([[150,0]]))
```



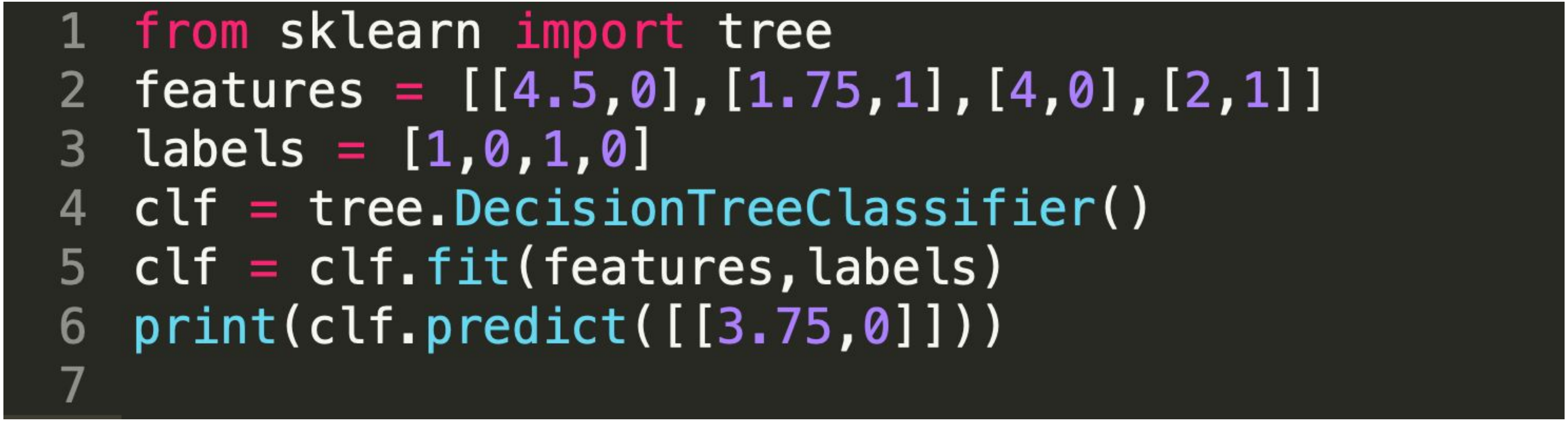
# ML Problems 2-3

Basic series of Classification problems

# Classification with Scikit Learn Problem #2

- 1. *Build a program that utilizes a classifier with scikit-learn to predict if a computer is a Mac or Windows based on two features: its weight and color (use your fruit problem example for inspiration) - For predicting, Mac is 0, Windows is 1*
    - Windows has a larger weight (~ 4 lbs), Mac has a smaller weight around (~ 2 lbs)
    - Mac has a color of white which corresponds to 1, while Windows has a color of black which corresponds to 0
- Here are your features and labels and make sure to fit them using a classifier
    - Features: [[4.5,0],[1.75,1],[4,0],[2,1]] Labels: [1,0,1,0]
- Save the program as computerprob1.py to desktop, then run
    - Predict the OS of a computer with weight of 3.75 pounds and a black color [3.75,0]

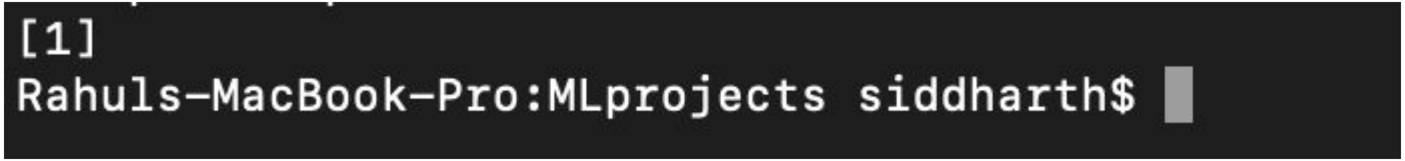
# Solution (Problem #2)



```python
from sklearn import tree
features = [[4.5,0],[1.75,1],[4,0],[2,1]]
labels = [1,0,1,0]
clf = tree.DecisionTreeClassifier()
clf = clf.fit(features,labels)
print(clf.predict([[3.75,0]]))

```

Output:

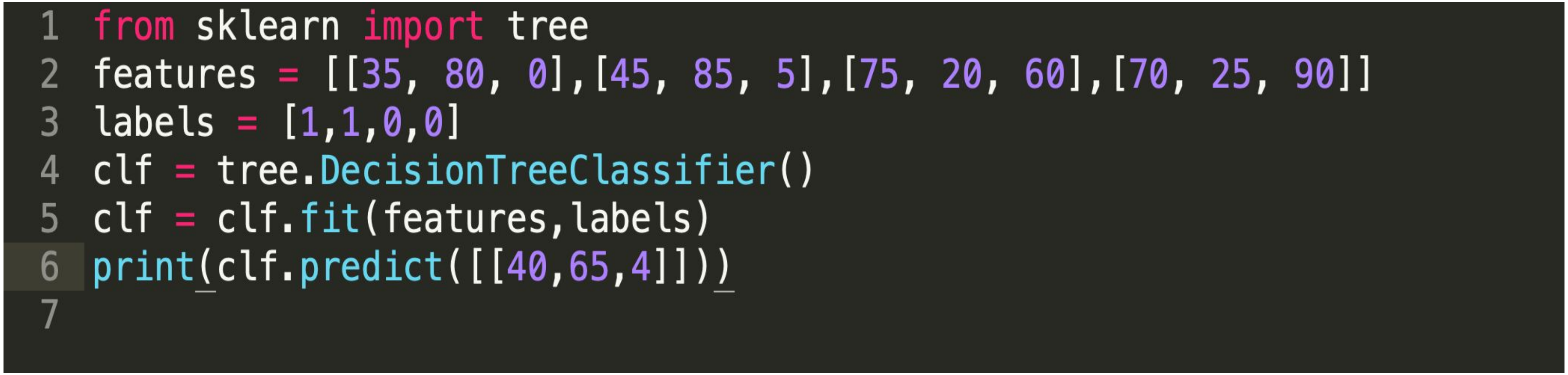


```
[1]
Rahuls-MacBook-Pro:MLprojects siddharth$ 
```

# Classification with Scikit-learn Problem #3

- 2. *Build a program that utilizes a classifier with scikit-learn to predict if a day has sunny or rainy weather based on* ***three*** *features: the humidity, the air pressure, and the precipitation: a rainy day is 0, sunny day is 1*
    - A sunny day has a humidity under 50%, and a rainy day has a humidity greater than 50%
    - A rainy day has a low air pressure (<30), and a sunny day has a high air pressure (~70)
    - A rainy day has precipitation greater than 40%, while a sunny day has under 10% precipitation
- Features are formatted as [humidity, air pressure, precipitation]
- Here are your features and labels:
    - Features - [[35, 80, 0],[45, 85, 5],[75, 20, 60],[70, 25, 90]]     Labels - [1,1,0,0]
- Predict the weather of a day with 40% humidity, 65 air pressure, and 4% precip.

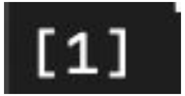
# Solution Problem #3



```python
from sklearn import tree
features = [[35, 80, 0],[45, 85, 5],[75, 20, 60],[70, 25, 90]]
labels = [1,1,0,0]
clf = tree.DecisionTreeClassifier()
clf = clf.fit(features,labels)
print(clf.predict([[40,65,4]]))

```

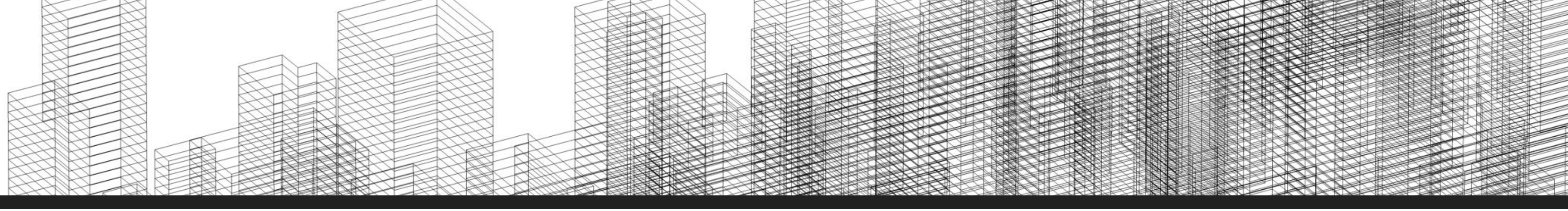
Output:



```
[1]
```

# Some notes

- These past two problems were quite easy, as we were just changing our features and labels
- In true Machine Learning, we don't *create* our features and labels, we use training and testing data sets
- The training set is what we apply to the model for supervised learning
- We use the testing set to test the accuracy of our model
- By using training sets and testings set, we can build an ML pipeline
- Having a "useless" feature can hurt our classifier accuracy
- We also don't want to use redundant features



# ML Classifiers Tutorial

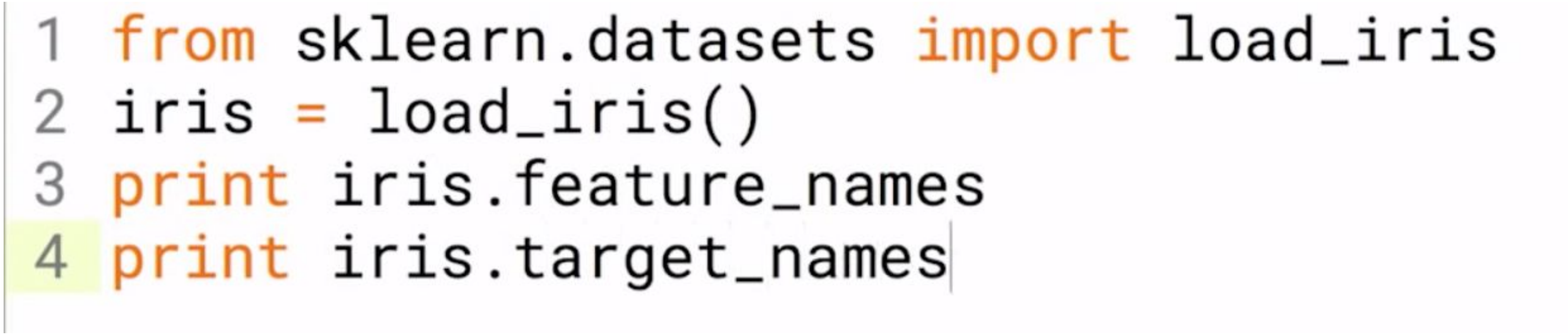
Decision Trees, Iris Datasets, and different types of Classifiers

# Visualizing decision trees and using datasets

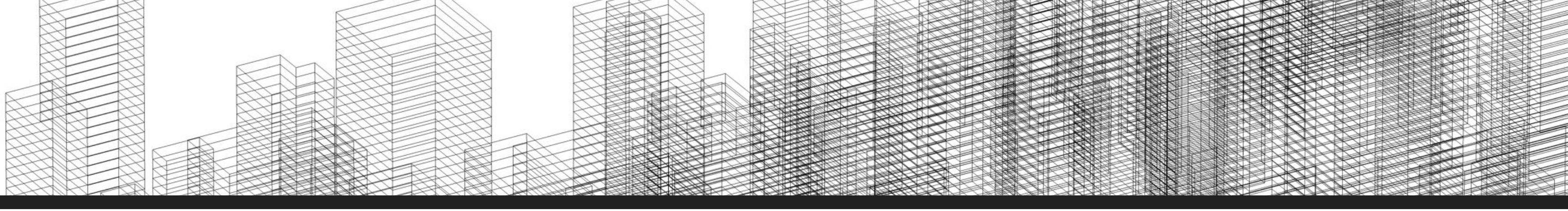
- We originally used a decision tree in our first fruit problem
- Now we can get a chance to visualize how it works while also learning about datasets and how to train your model

# How to load datasets

- To load a dataset like iris, we simple import it from the default database of datasets available in scikit-learn
- We also need metadata which tells us the name of features and labels in the dataset



```
from sklearn.datasets import load_iris
iris = load_iris()
print iris.feature_names
print iris.target_names
```



# Intro to ML Pipelines

Simple Classification Pipelines and accuracy

# What is an ML Pipeline?

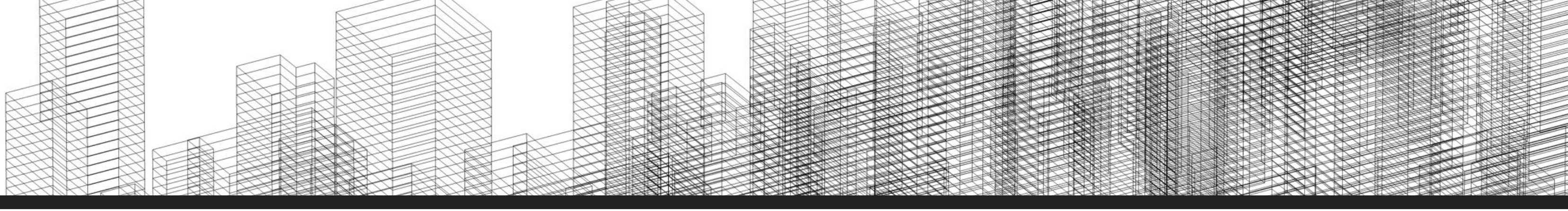
- A pipeline is just when you're chaining together different operations in the classification.
  - When we trained the classifier using the training set, and then tested using the test set. This was a simple pipeline.
- The accuracy of a pipeline can be determined using scikit-learn
- Our goal is to maximize the accuracy of an ML pipeline

# The Classifier as a function pt. 1

- Originally, we talked about our classifier as a box of rules
- We can also think about it as a function
- A function is a mapping of input to output values
- f(x) = y → f(x) is our feature, y is our label
- We want our function to be able to *learn* from data
- We don’t want to write our function, but rather a supervised learning algorithm should be able to learn it from training data
- We will cover this more next time

# Takeaways from today

- In under 10 lines of code, you can build a powerful decision tree classifier
- We utilize training and testing datasets to train our model and apply it to labeled examples
- A decision tree is a sequence of pathways which look for certain features in order to predict a label
- A ML pipeline chains together different operations in the classification.
- The classifier can also be thought of as a function of features and labels



# Club Information

Some Quick Information regarding the club

# Links and Housekeeping

- Please spare a few moments to take this short survey regarding the content for the rest of the year: https://goo.gl/forms/Ni2RIqqgQ988Zdd23
- Here is the link to the google drive for our club meeting resources and ML library information. https://tinyurl.com/y92alrbb
- 
- Join our Slack for receiving messages and daily updates on club meetings and information. https://tinyurl.com/ybcbvea2
- Let’s divide into teams for the contest:)

# Next Meeting

- Next time, we will be doing classification with Support Vector Machines (SVM) while learning more about ML pipelines
- Before winter break, we will jump into regression with scikit-learn
- Hopefully, everyone should have all the software downloaded by now
- In the next few meetings we will jump right into the concepts without any review so please be prepared
  - We can try predicting SAT scores and house prices later on as regression is even more powerful
- Thanks for a great first third of the year:)

# Sources

- All credit goes to Josh Gordon (Youtube ML recipes)
- Google ML Crash Course
- Scikit-learn Classification Tutorial
- Wikipedia Iris Datasets
- Stanford Coursera ML