10-601 Introduction to Machine Learning

Machine Learning Department
School of Computer Science
Carnegie Mellon University

k-Nearest Neighbors
+
Model Selection

Matt Gormley
Lecture 5
Jan. 29, 2020

Q&A

Q: Why don't my entropy calculations match those on the slides?

A: H(Y) is conventionally reported in "bits" and computed using log base 2.
e.g., $H(Y) = - P(Y=0) \log_2 P(Y=0) - P(Y=1) \log_2 P(Y=1)$

Q: Why is entropy based on a sum of p(.) log p(.) terms?

A: We don't have time for a full treatment of why it *has* to be this, but we can develop the right intuition with a few examples…

Q&A

Q: How do we deal with ties in k-Nearest Neighbors (e.g. even k or equidistant points)?

A: I would ask you all for a good solution!

Q: How do we define a distance function when the features are categorical (e.g. weather takes values {sunny, rainy, overcast})?

A: Step 1: Convert from categorical attributes to numeric features (e.g. binary)

Step 2: Select an appropriate distance function (e.g. Hamming distance)

Reminders

- Homework 2: Decision Trees
  - Out: Wed, Jan. 22
  - Due: Wed, Feb. 05 at 11:59pm
- Today's Poll:
  - http://p5.mlcourse.org

Moss Cheat Checker

What is Moss?

- Moss (Measure Of Software Similarity): is an automatic system for determining the similarity of programs. To date, the main application of Moss has been in detecting plagiarism in programming classes.
- Moss reports:
  - The Andrew IDs associated with the file submissions
  - The number of lines matched
  - The percent lines matched
  - Color coded submissions where similarities are found

What is Moss?

At first glance, the submissions may look different

```python
# Python program to find ordered words
import requests

# Scrapes the words from the URL below and stores
# them in a list
def getWords():

# contains about 2500 words
    url = "http://www.puzzlers.org/pub/wordlists/unixdict.txt"
    fetchData = requests.get(url)

# extracts the content of the webpage
    wordList = fetchData.content

# decodes the UTF-8 encoded text and splits the
# string to turn it into a list of words
    wordList = wordList.decode("utf-8").split()

    return wordList


# function to determine whether a word is ordered or not
def isOrdered():

# fetching the wordList
    collection = getWords()

# since the first few of the elements of the
# dictionary are numbers, getting rid of those
# numbers by slicing off the first 17 elements
    collection = collection[16:]
    word = ''

    for word in collection:
        result = 'Word is ordered'
        i = 0
        l = len(word) - 1

        if (len(word) < 3): # skips the 1 and 2 lettered strings
            continue

        # traverses through all characters of the word in pairs
        while i < l:
            if (ord(word[i]) > ord(word[i+1])):
                result = 'Word is not ordered'
                break
            else:
                i += 1

    # only printing the ordered words
        if (result == 'Word is ordered'):
            print(word,': ',result)


# execute isOrdered() function
if __name__ == '__main__':
    isOrdered()
```

```python
import requests

def Ordered():
    coll = getWs()
    coll = coll[16:]
    word = ''
    for word in coll:
        r = 'Word is ordered'
        a = 0
        length = len(word) - 1
        if (len(word) < 3):
            continue
        while a < length:
            if (ord(word[a]) > ord(word[a+1])):
                r = 'Word is not ordered'
                break
            else:
                a += 1
        if (r == 'Word is ordered'):
            print(word,': ',r)

def getWs():
    url = "http://www.puzzlers.org/pub/wordlists/unixdict.txt"
    fetch = requests.get(url)
    words = fetch.content
    words = words.decode("utf-8").split()
    return words

if __name__ == '__main__':
    Ordered()
```

What is Moss?

Moss can quickly find the similarities

OVERFITTING
(FOR DECISION TREES)

Decision Tree Generalization

Question:

Which of the following would generalize best to unseen examples?

A. Small tree with low training accuracy

B. Large tree with low training accuracy

C. Small tree with high training accuracy

D. Large tree with high training accuracy

Answer:

Overfitting and Underfitting

Underfitting

- The model…
  - is too simple
  - is unable captures the trends in the data
  - exhibits too much bias
- *Example*: majority-vote classifier (i.e. depth-zero decision tree)
- *Example*: a toddler (that has **not** attended medical school) attempting to carry out medical diagnosis

Overfitting

- The model…
  - is too complex
  - is fitting the noise in the data
  - or fitting random statistical fluctuations inherent in the "sample" of training data
  - does not have enough bias
- *Example*: our "memorizer" algorithm responding to an "orange shirt" attribute
- *Example*: medical student who simply memorizes patient case studies, but does not understand how to apply knowledge to new patients

Overfitting

- Consider a hypothesis *h* its…
  - …error rate over all training data: $error(h, D_{train})$
  - …error rate over all test data: $error(h, D_{test})$

Slide adapted from Tom Mitchell

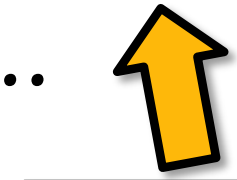
Overfitting

- Consider a hypothesis *h* its…
  - …error rate over all training data: $error(h, D_{train})$
  - …error rate over all test data: $error(h, D_{test})$
  - …true error over all data: $error_{true}(h)$
- We say h overfits the training data if…

$$error_{true}(h) > error(h, D_{train})$$

- Amount of overfitting =

$$error_{true}(h) - error(h, D_{train})$$



In practice, $error_{true}(h)$ is **unknown**

Slide adapted from Tom Mitchell

Overfitting in Decision Tree Learning

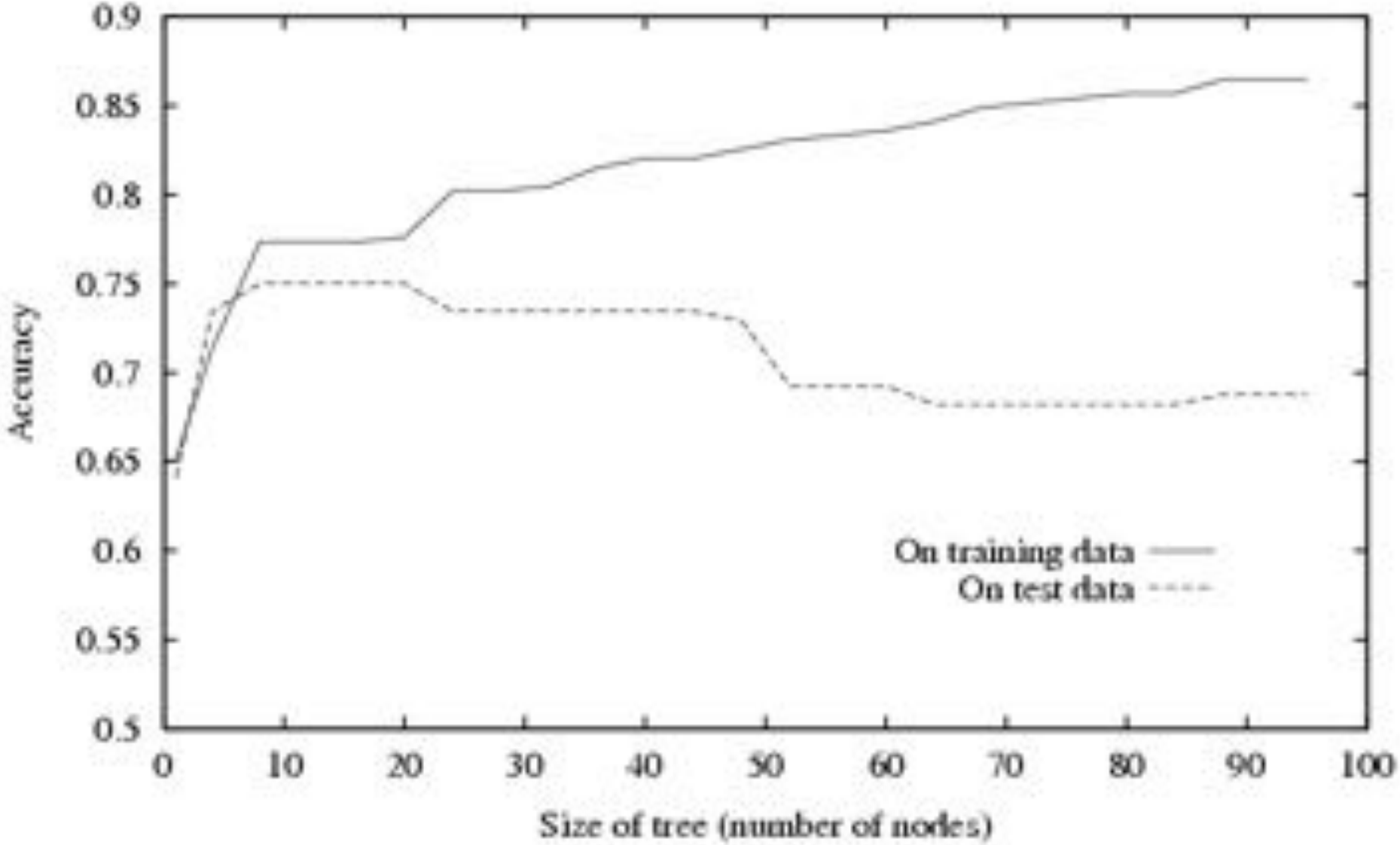


Figure from Tom Mitchell

How to Avoid Overfitting?

For Decision Trees…

1. Do not grow tree beyond some **maximum depth**
2. Do not split if splitting criterion (e.g. mutual information) is **below some threshold**
3. Stop growing when the split is **not statistically significant**
4. Grow the entire tree, then **prune**

Reduced-Error Pruning

Split data into *training* and *validation* set

Create tree that classifies *training* set correctly

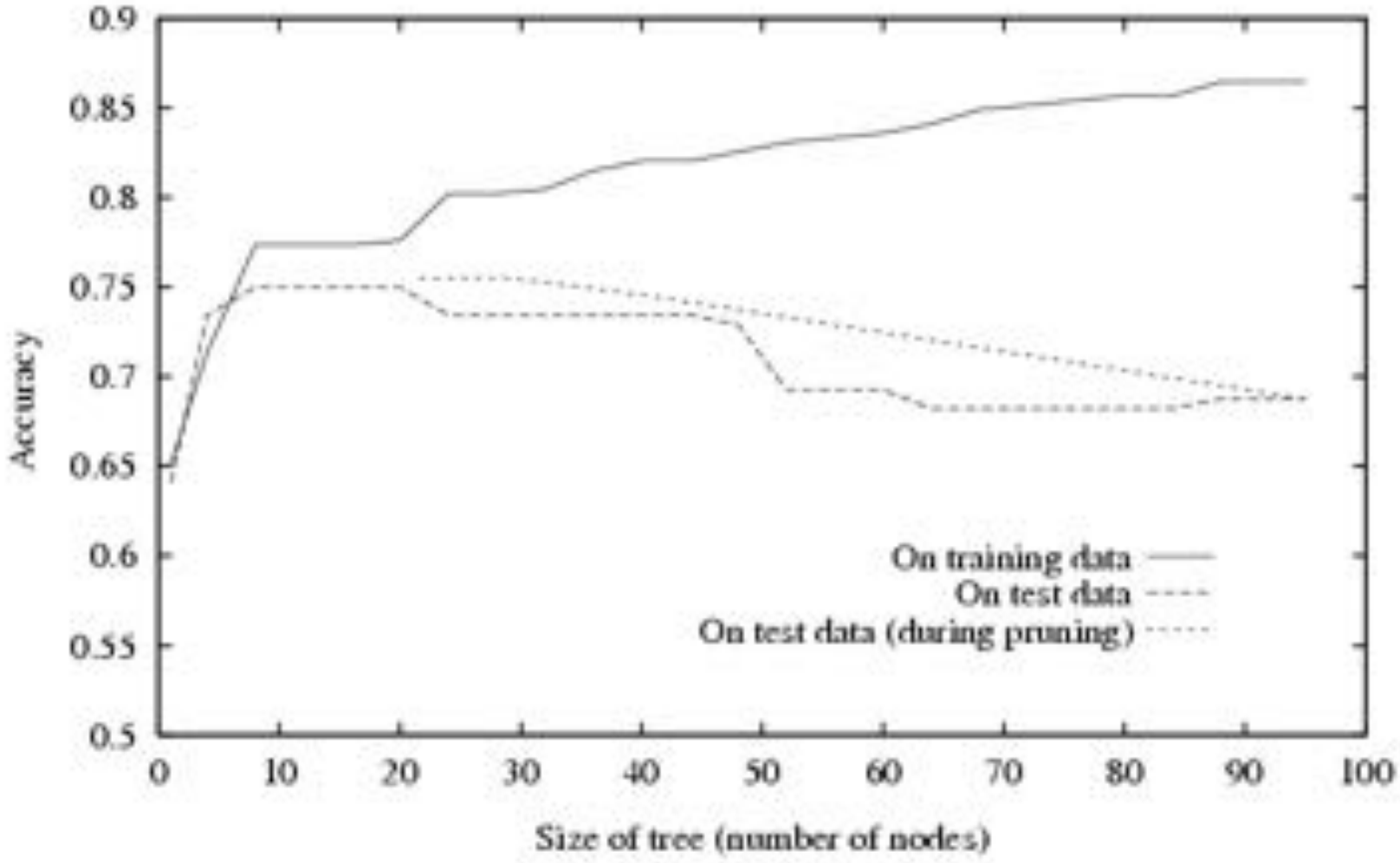
Do until further pruning is harmful:

1. Evaluate impact on *validation* set of pruning each possible node (plus those below it)
2. Greedily remove the one that most improves *validation* set accuracy

- produces smallest version of most accurate subtree
- What if data is limited?

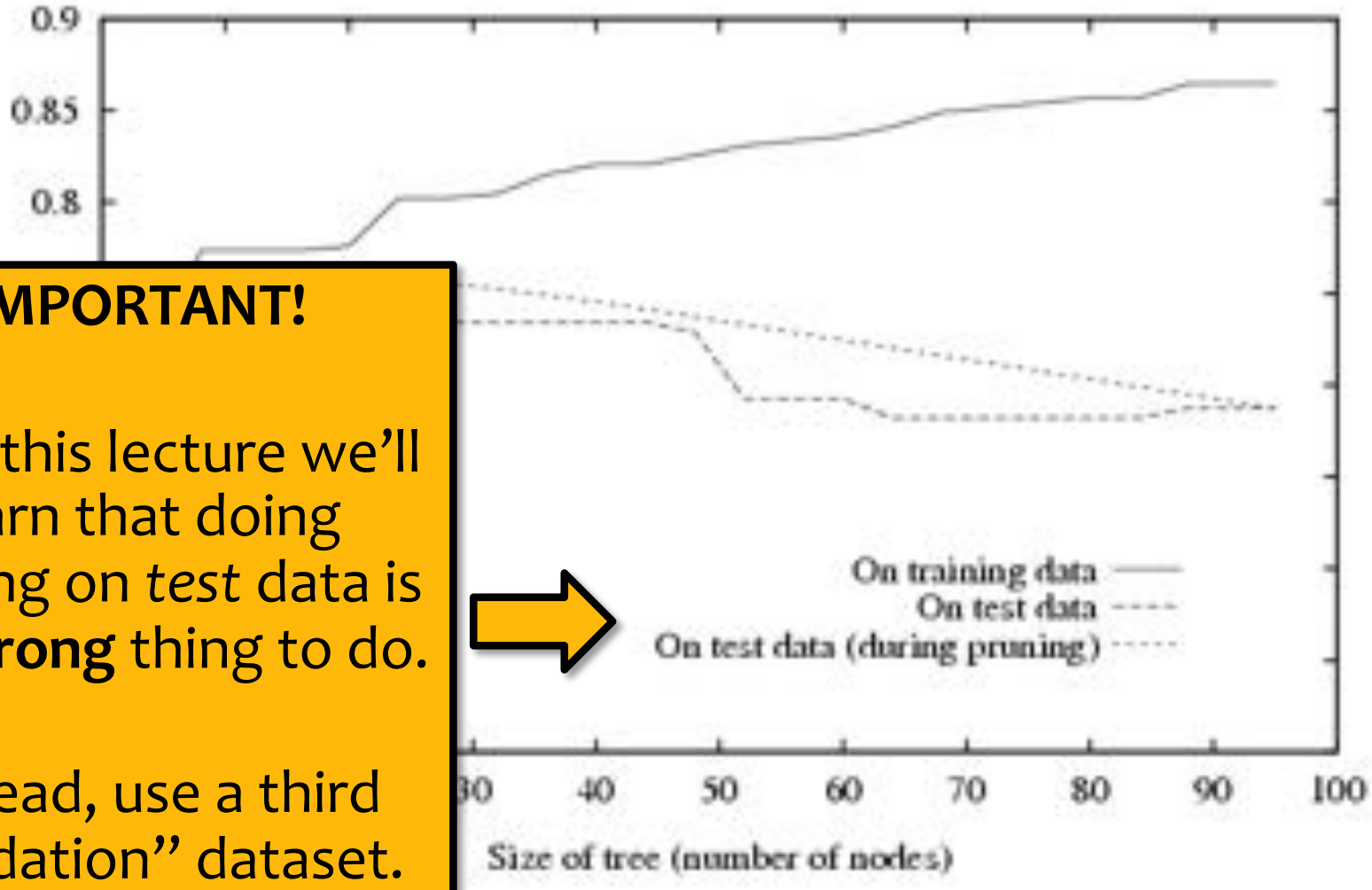
Slide from Tom Mitchell

Effect of Reduced-Error Pruning



Slide from Tom Mitchell

Effect of Reduced-Error Pruning



IMPORTANT!

Later this lecture we'll learn that doing pruning on *test* data is the **wrong** thing to do.

Instead, use a third "validation" dataset.

Slide from Tom Mitchell

Decision Trees (DTs) in the Wild

- DTs are one of the most popular classification methods for practical applications
  - Reason #1: The learned representation is **easy to explain** a non-ML person
  - Reason #2: They are **efficient** in both computation and memory
- DTs can be applied to a wide variety of problems including **classification**, **regression**, **density estimation**, etc.
- **Applications of DTs** include…
  - medicine, molecular biology, text classification, manufacturing, astronomy, agriculture, and many others
- **Decision Forests** learn many DTs from random subsets of features; the result is a very powerful example of an **ensemble method** (discussed later in the course)

DT Learning Objectives

You should be able to…

1. Implement Decision Tree training and prediction
2. Use effective splitting criteria for Decision Trees and be able to define entropy, conditional entropy, and mutual information / information gain
3. Explain the difference between memorization and generalization [CIML]
4. Describe the inductive bias of a decision tree
5. Formalize a learning problem by identifying the input space, output space, hypothesis space, and target function
6. Explain the difference between true error and training error
7. Judge whether a decision tree is "underfitting" or "overfitting"
8. Implement a pruning or early stopping method to combat overfitting in Decision Tree learning

K-NEAREST NEIGHBORS

Classification

Chalkboard:

- Binary classification
- 2D examples
- Decision rules / hypotheses

k-Nearest Neighbors

Chalkboard:

- Nearest Neighbor classifier
- KNN for binary classification

KNN: Remarks

Distance Functions:

- KNN requires a **distance function**

$$g : \mathbb{R}^M \times \mathbb{R}^M \rightarrow \mathbb{R}$$

- The most common choice is **Euclidean distance**

$$g(\mathbf{u}, \mathbf{v}) = \sqrt{\sum_{m=1}^{M} (u_m - v_m)^2}$$

- But other choices are just fine (e.g. **Manhattan distance**)

$$g(\mathbf{u}, \mathbf{v}) = \sum_{m=1}^{M} |u_m - v_m|$$

KNN: Remarks

In-Class Exercises

1. How can we handle ties for even values of k?

2. What is the inductive bias of KNN?

Answer(s) Here:

KNN: Remarks

In-Class Exercises

1. How can we handle ties for even values of k?

2. What is the inductive bias of KNN?

Answer(s) Here:

1)

- Consider another point
- Remove farthest of k points
- Weight votes by distance
- Consider another distance metric

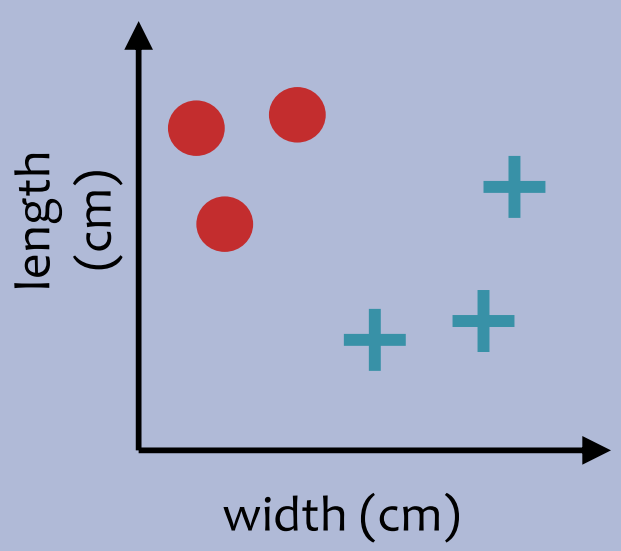
2)

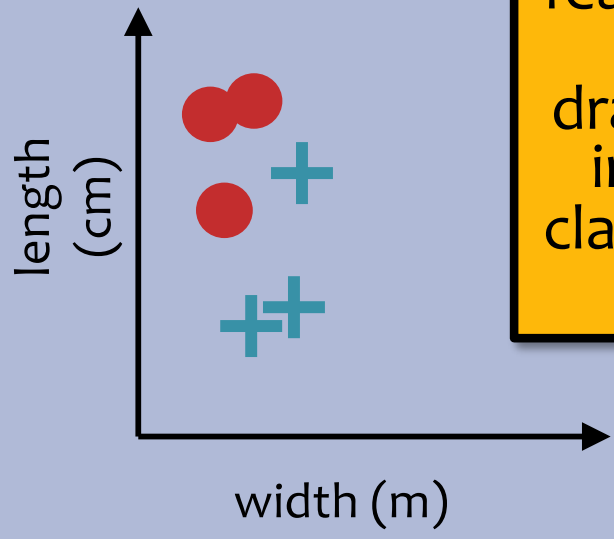
KNN: Remarks

Inductive Bias:

1. Similar points should have similar labels
2. All dimensions are created equally!

Example: two features for KNN





big problem: feature scale could dramatically influence classification results

KNN: Remarks

Computational Efficiency:

- Suppose we have N training examples, and each one has M features
- Computational complexity for the special case where k=1:

Task	Naive	k-d Tree
Train	O(1)	~ O(M N log N)
Predict (one test example)	O(MN)	~ O(2^M log N) on average

Problem: Very fast for small M, but very slow for large M

In practice: use stochastic approximations (very fast, and empirically often as good)

KNN: Remarks

Theoretical Guarantees:

Cover & Hart (1967)

Let h(x) be a Nearest Neighbor (k=1) binary classifier. As the number of training examples N goes to infinity…

$$\text{error}_{\text{true}}(h) < 2 \times \text{Bayes Error Rate}$$

"In this sense, it may be said that half the classification information in an infinite sample set is contained in the nearest neighbor."

very informally, Bayes Error Rate can be thought of as: '*the best you could possibly do*'

Decision Boundary Example

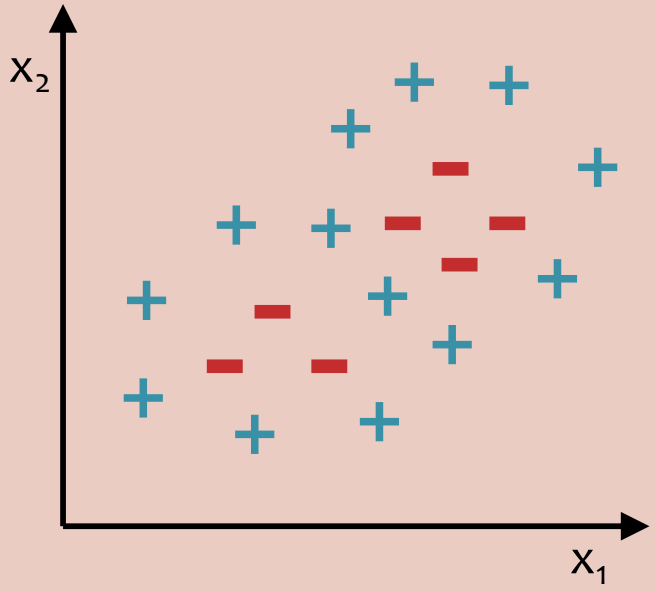
Dataset: Outputs {+,-}; Features x_1 and x_2

In-Class Exercise

Question 1:

A. Can a **k-Nearest Neighbor classifier with k=1** achieve **zero training error** on this dataset?

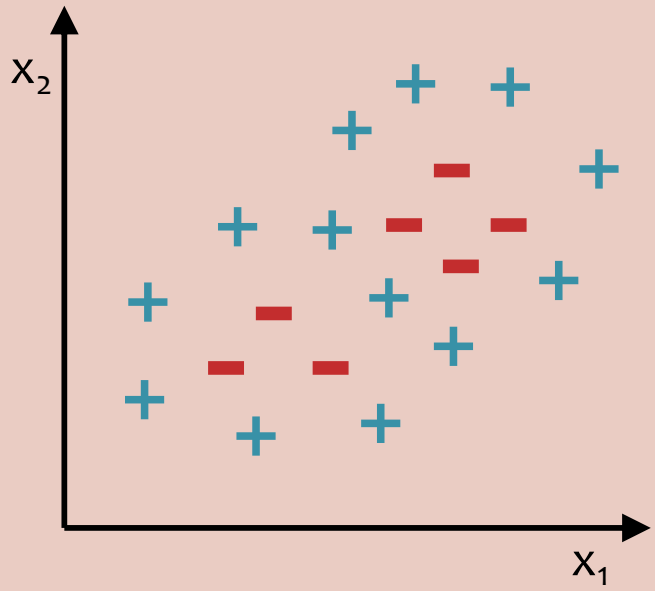
B. **If 'Yes'**, draw the learned decision boundary. **If 'No'**, why not?



Question 2:

A. Can a **Decision Tree classifier** achieve **zero training error** on this dataset?

B. **If 'Yes'**, draw the learned decision bound. **If 'No'**, why not?



KNN ON FISHER IRIS DATA

Fisher Iris Dataset

Fisher (1936) used 150 measurements of flowers from 3 different species: Iris setosa (0), Iris virginica (1), Iris versicolor (2) collected by Anderson (1936)

Species	Sepal Length	Sepal Width	Petal Length	Petal Width
0	4.3	3.0	1.1	0.1
0	4.9	3.6	1.4	0.1
0	5.3	3.7	1.5	0.2
1	4.9	2.4	3.3	1.0
1	5.7	2.8	4.1	1.3
1	6.3	3.3	4.7	1.6
1	6.7	3.0	5.0	1.7

Full dataset: https://en.wikipedia.org/wiki/Iris_flower_data_set

Fisher Iris Dataset

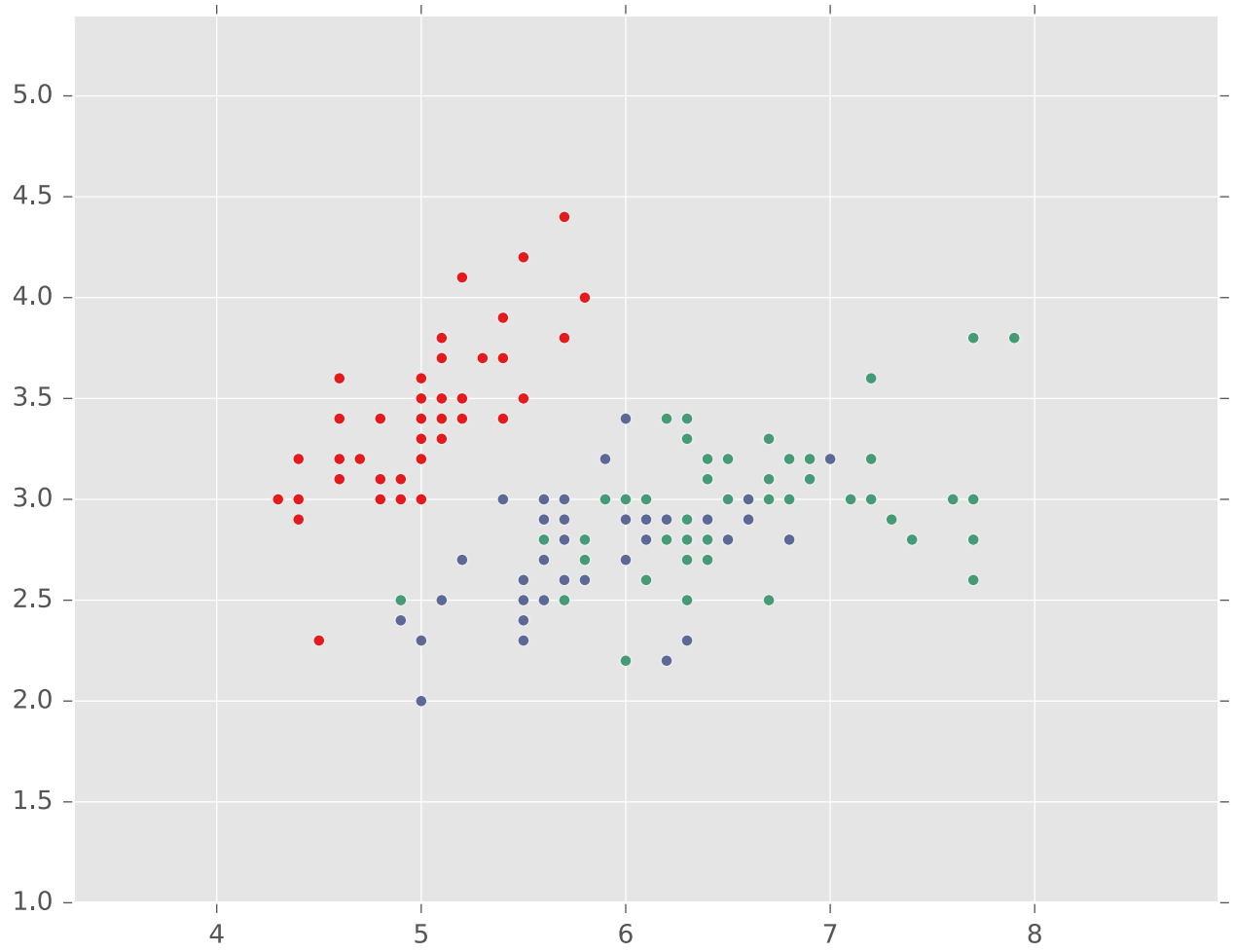
Fisher (1936) used 150 measurements of flowers from 3 different species: Iris setosa (0), Iris virginica (1), Iris versicolor (2) collected by Anderson (1936)

Species	Sepal Length	Sepal Width
0	4.3	3.0
0	4.9	3.6
0	5.3	3.7
1	4.9	2.4
1	5.7	2.8
1	6.3	3.3
1	6.7	3.0

Deleted two of the four features, so that input space is 2D

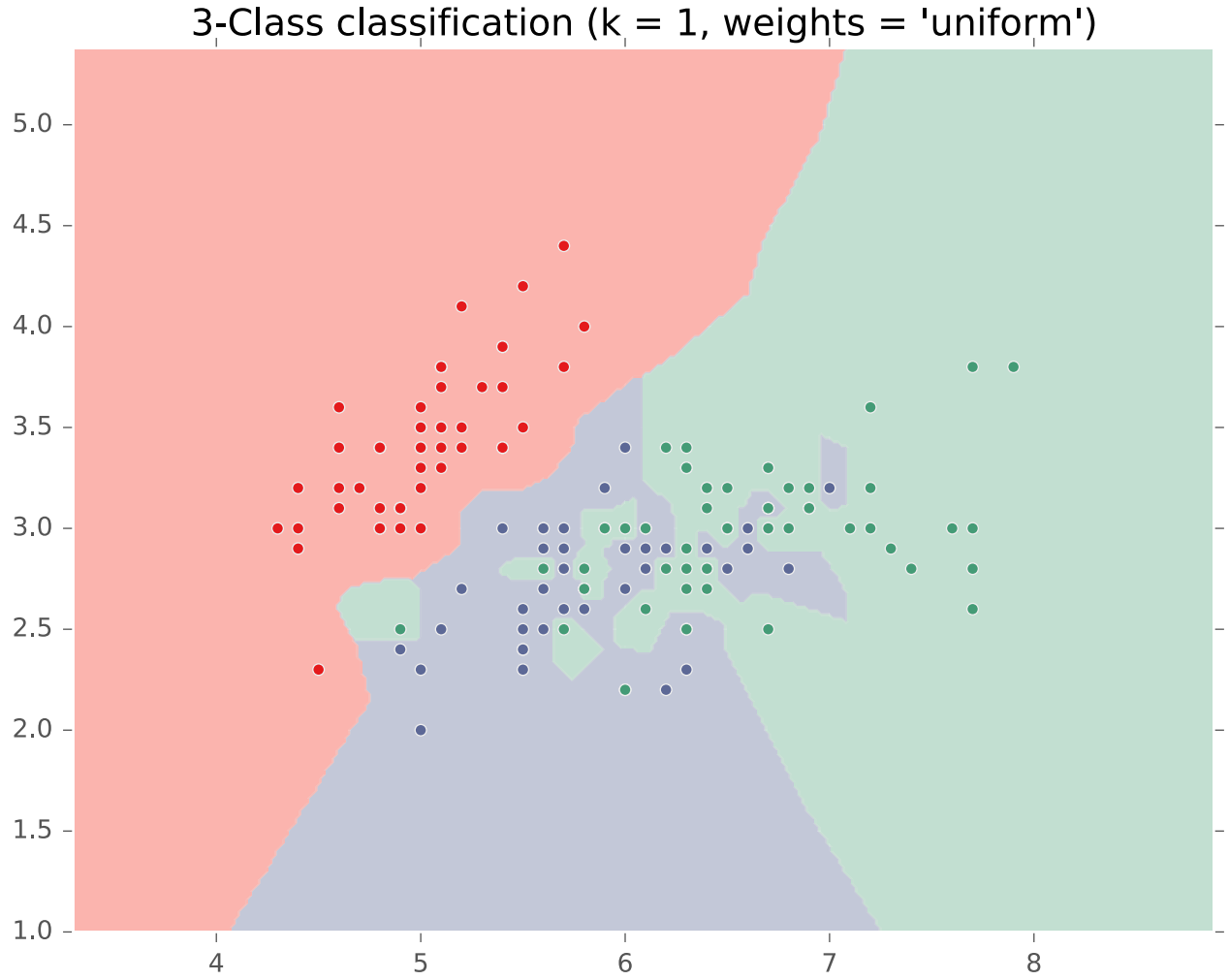
Full dataset: https://en.wikipedia.org/wiki/Iris_flower_data_set

KNN on Fisher Iris Data

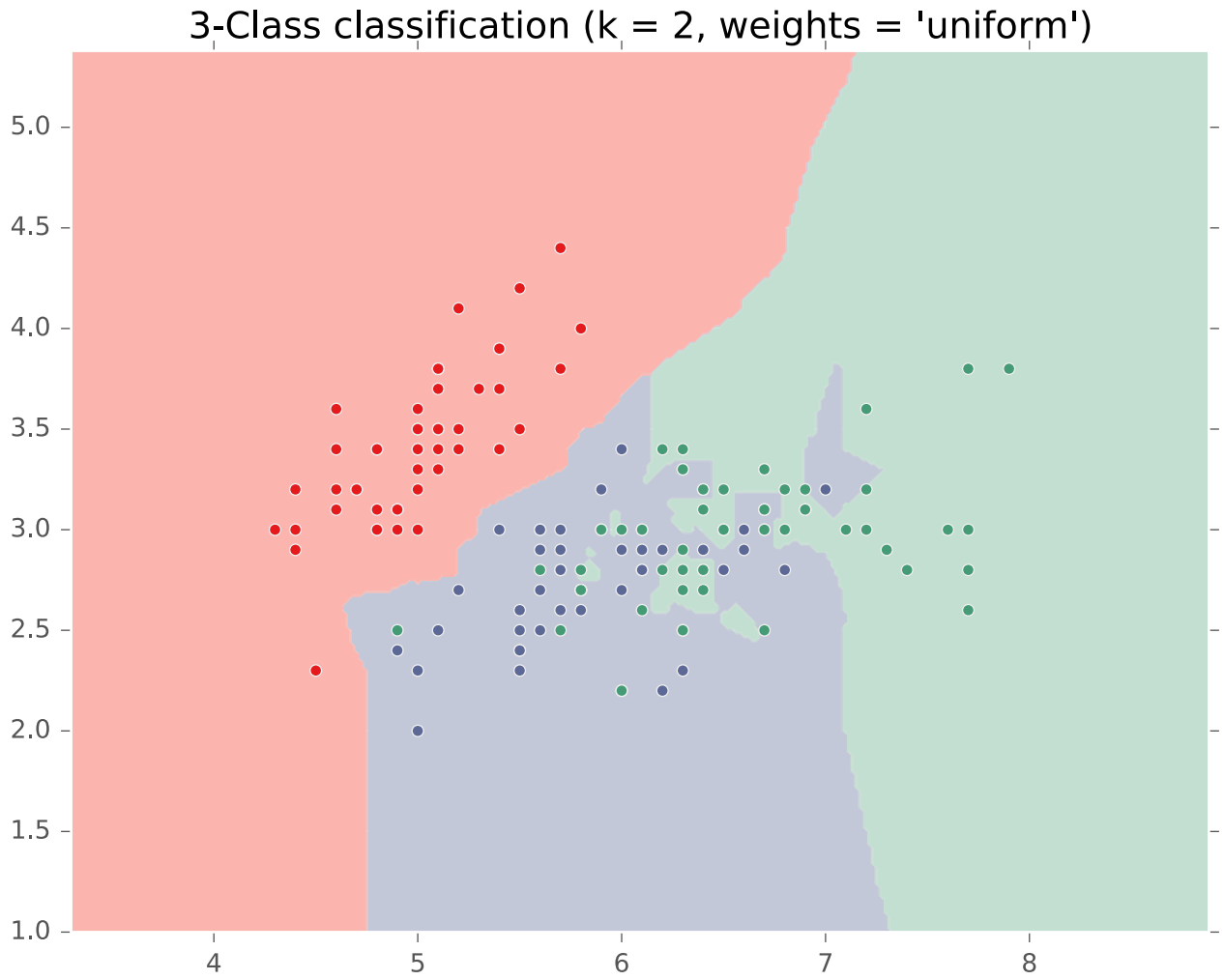


KNN on Fisher Iris Data

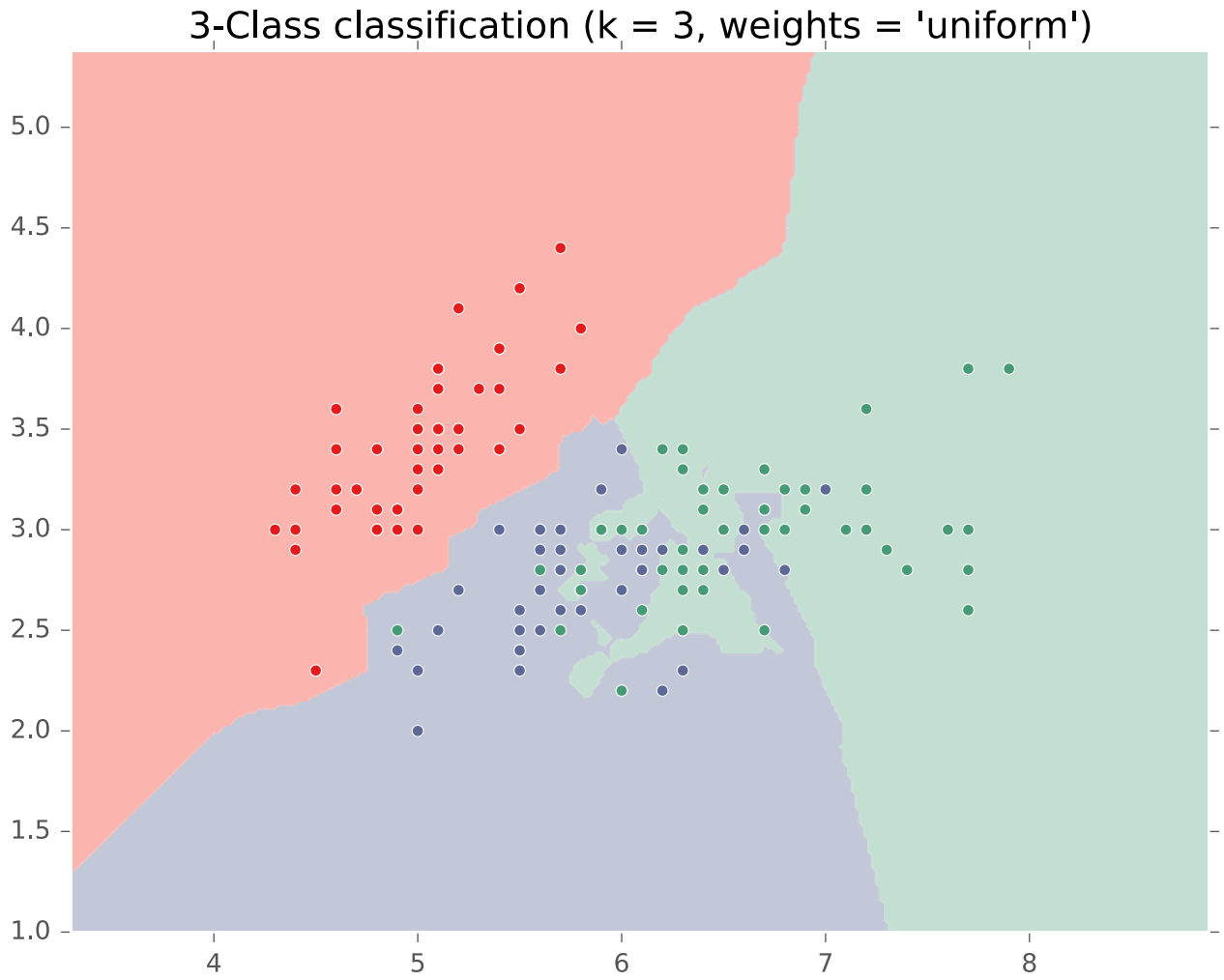
Special Case: Nearest Neighbor



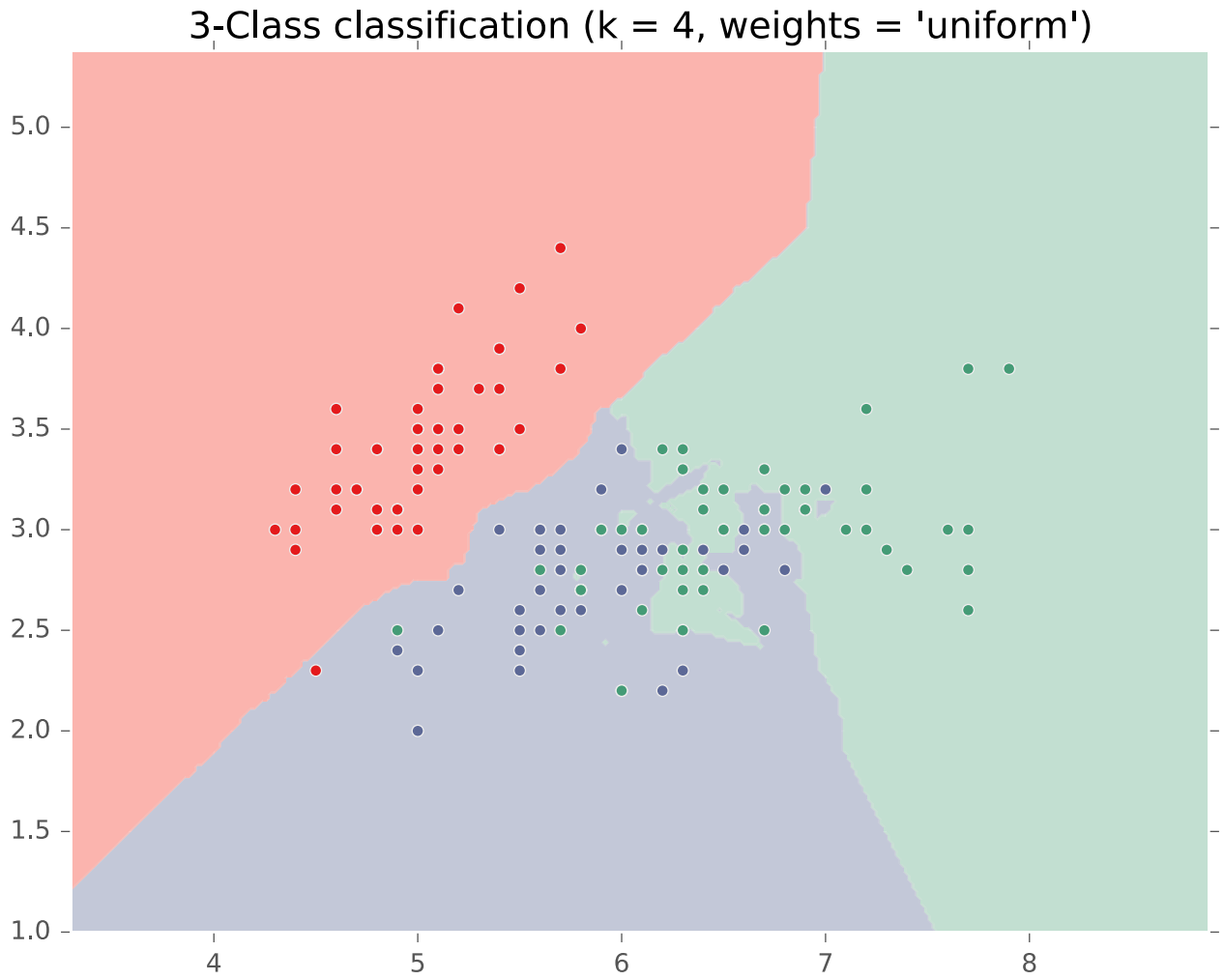
KNN on Fisher Iris Data



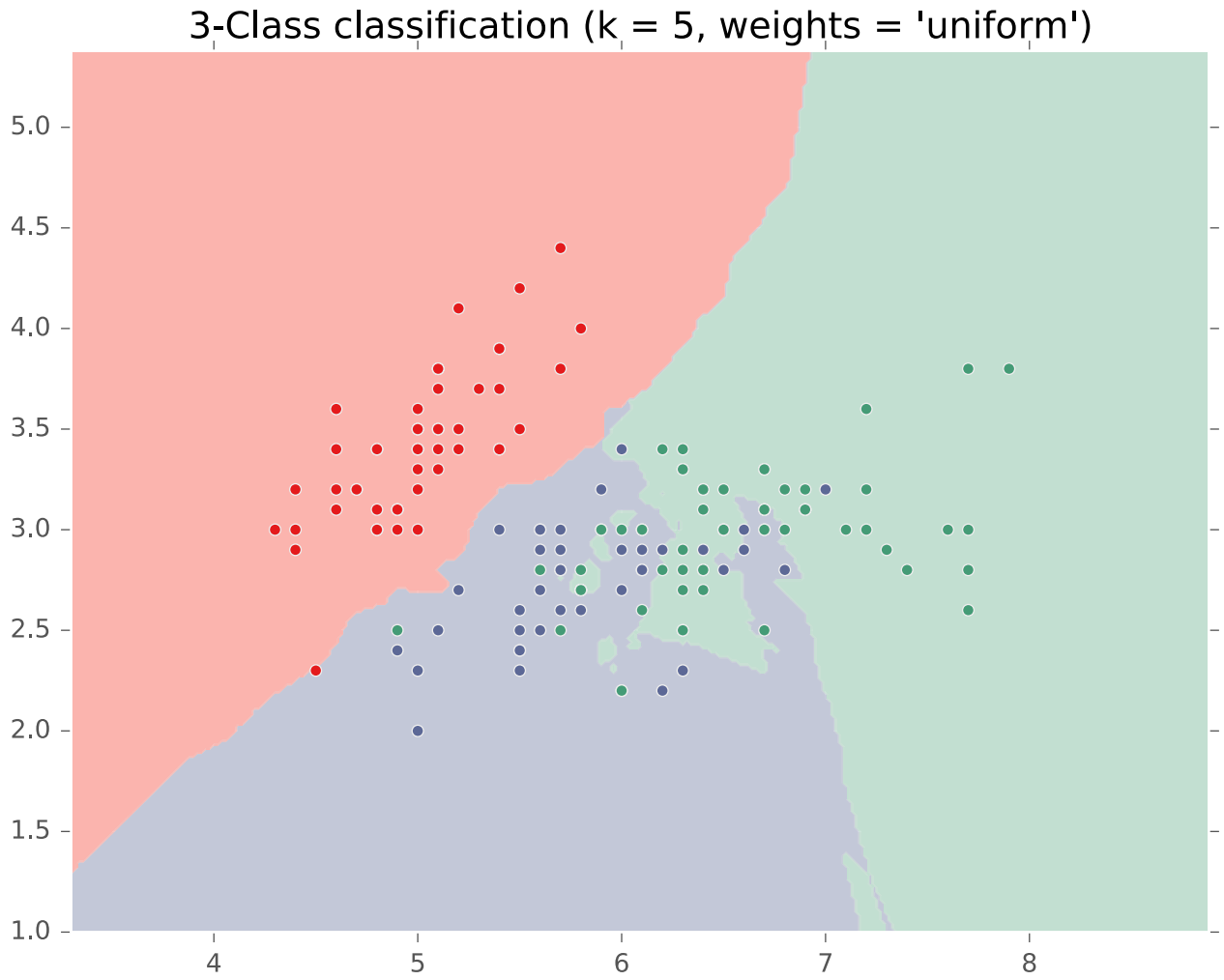
KNN on Fisher Iris Data



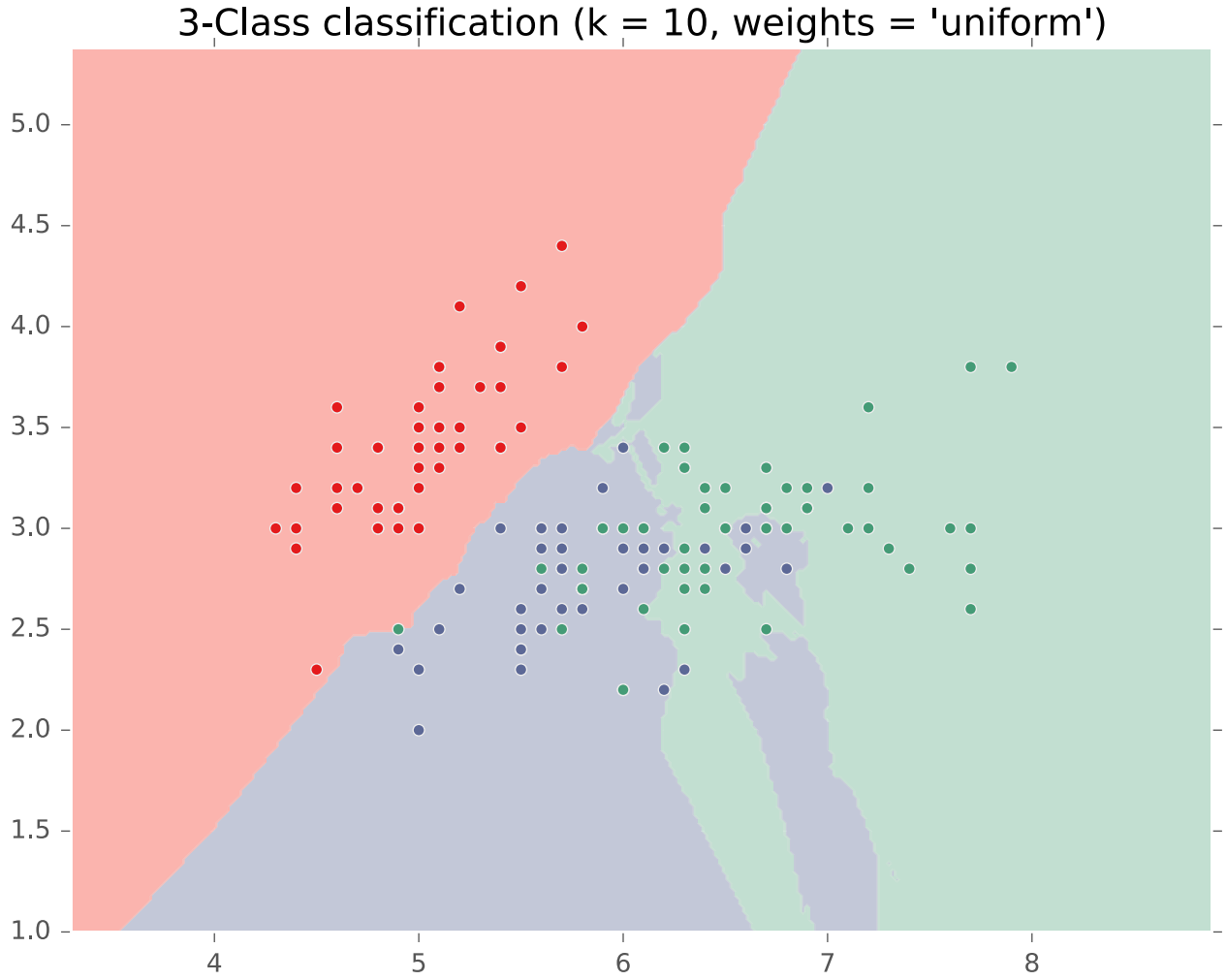
KNN on Fisher Iris Data



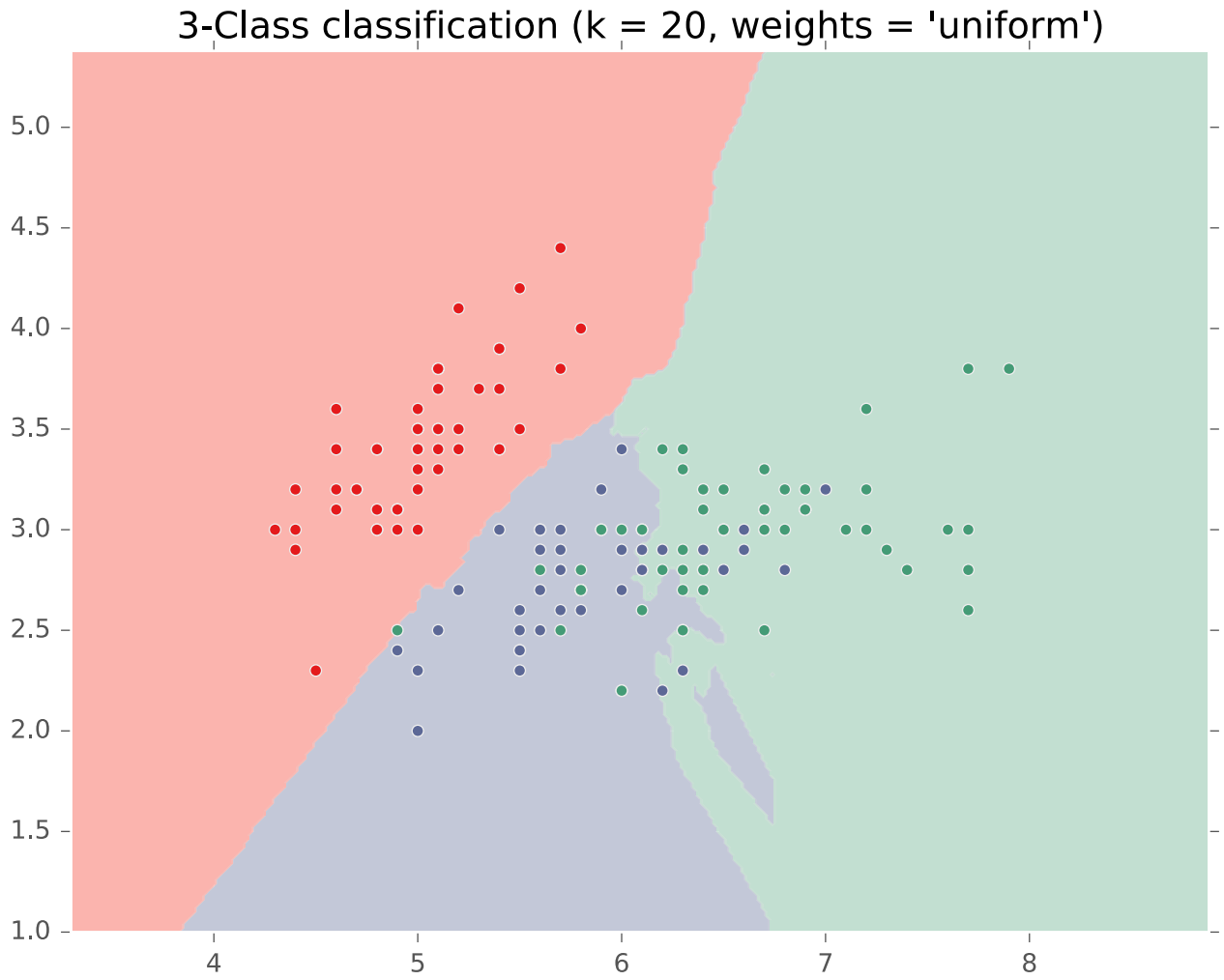
KNN on Fisher Iris Data



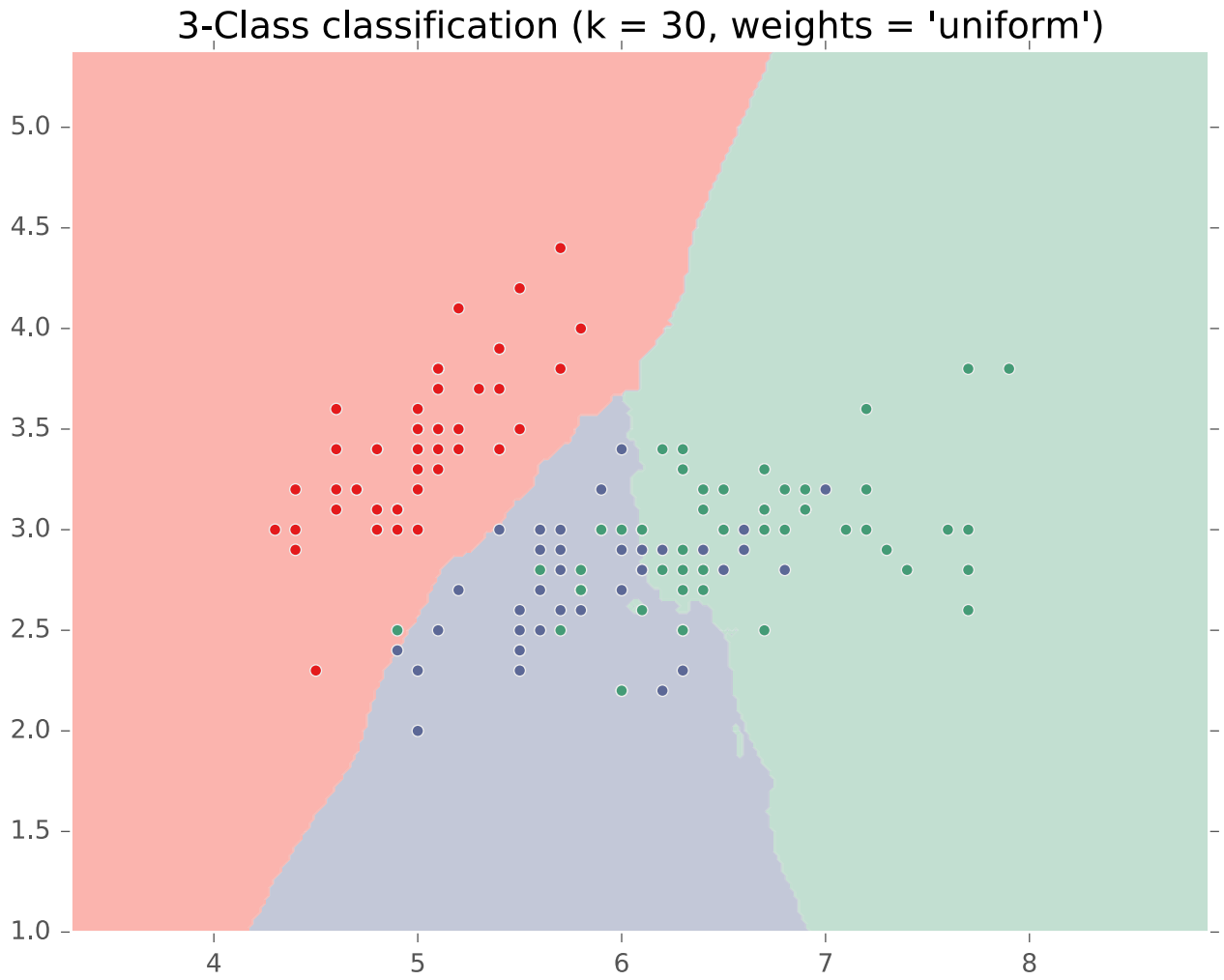
KNN on Fisher Iris Data



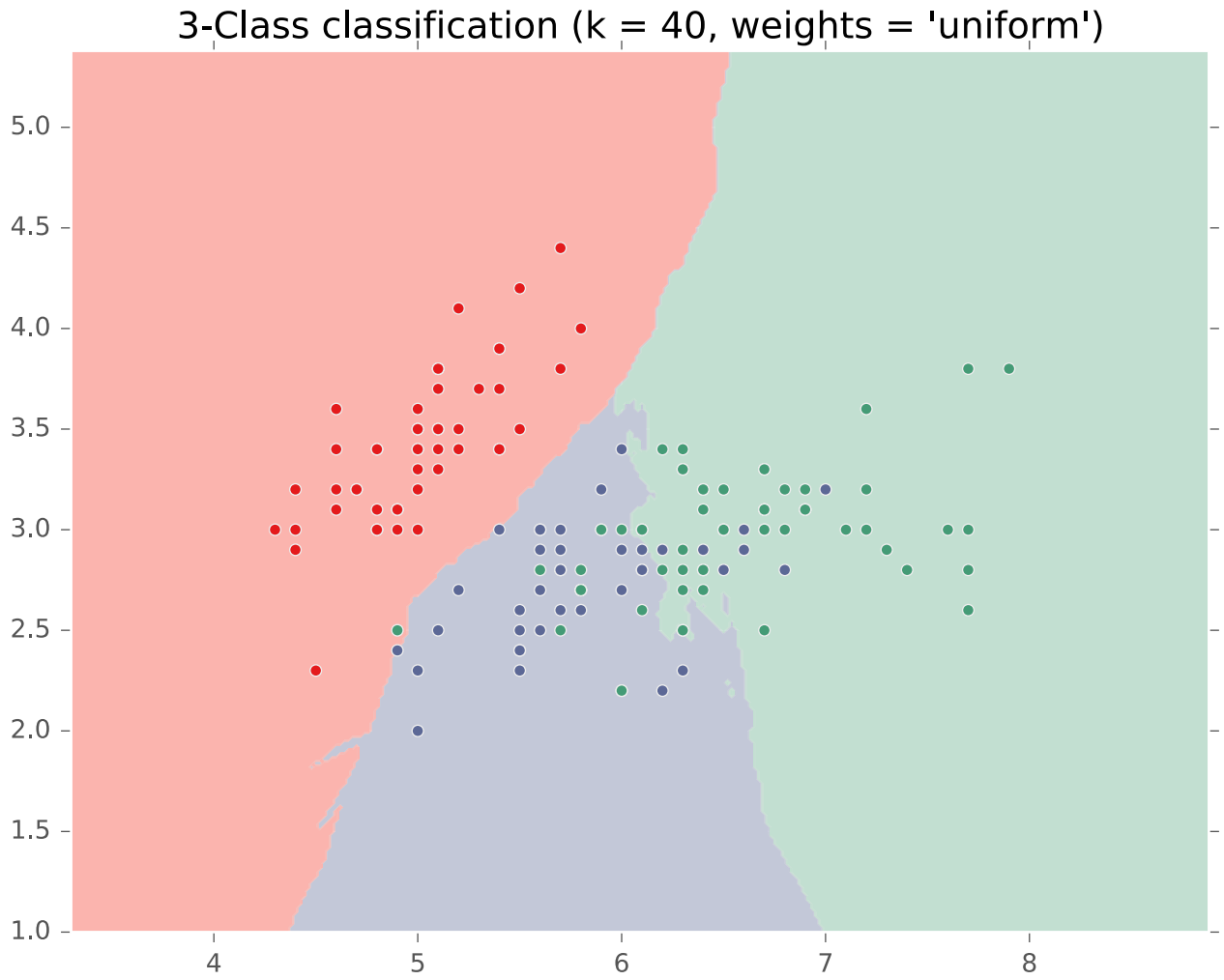
KNN on Fisher Iris Data



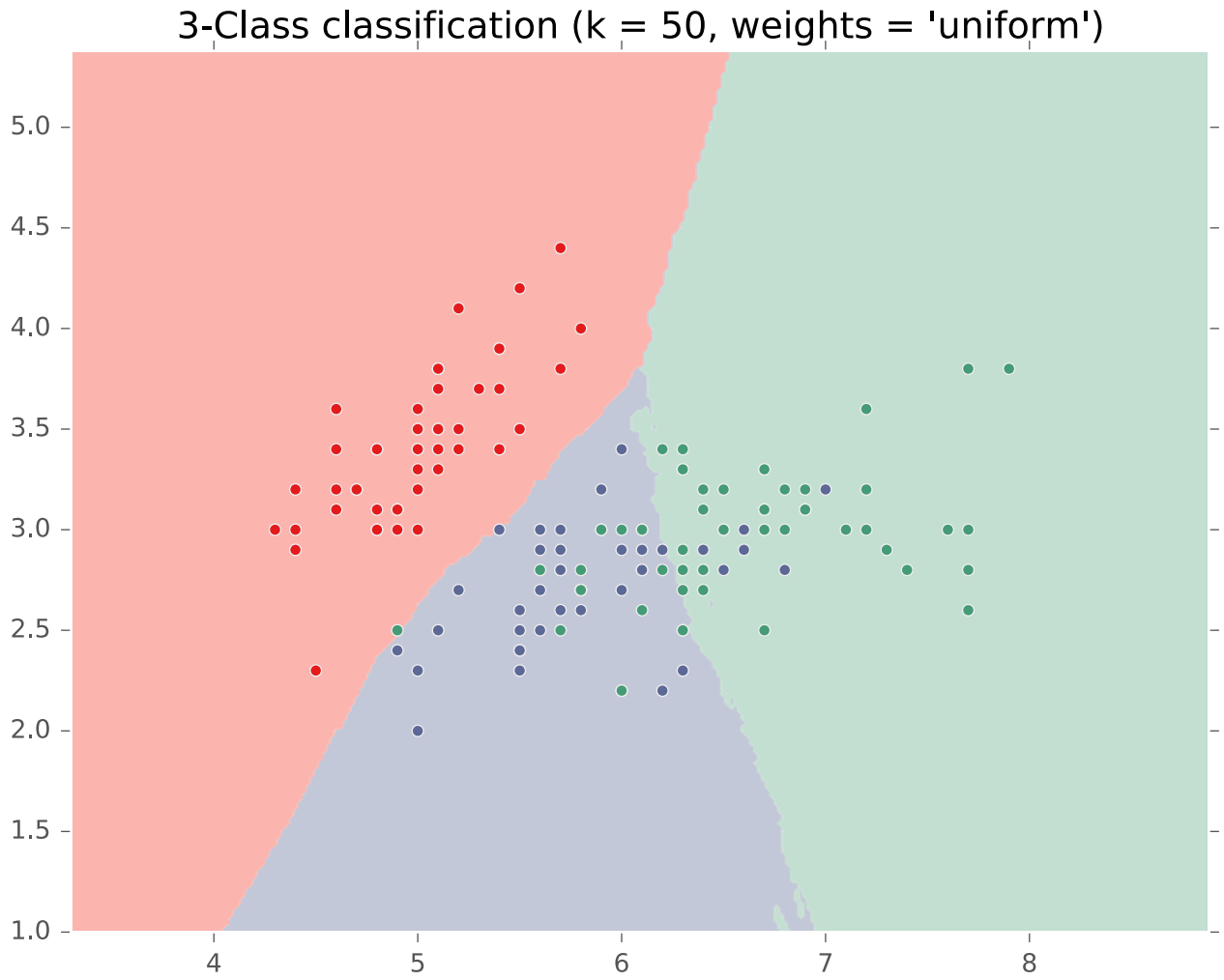
KNN on Fisher Iris Data



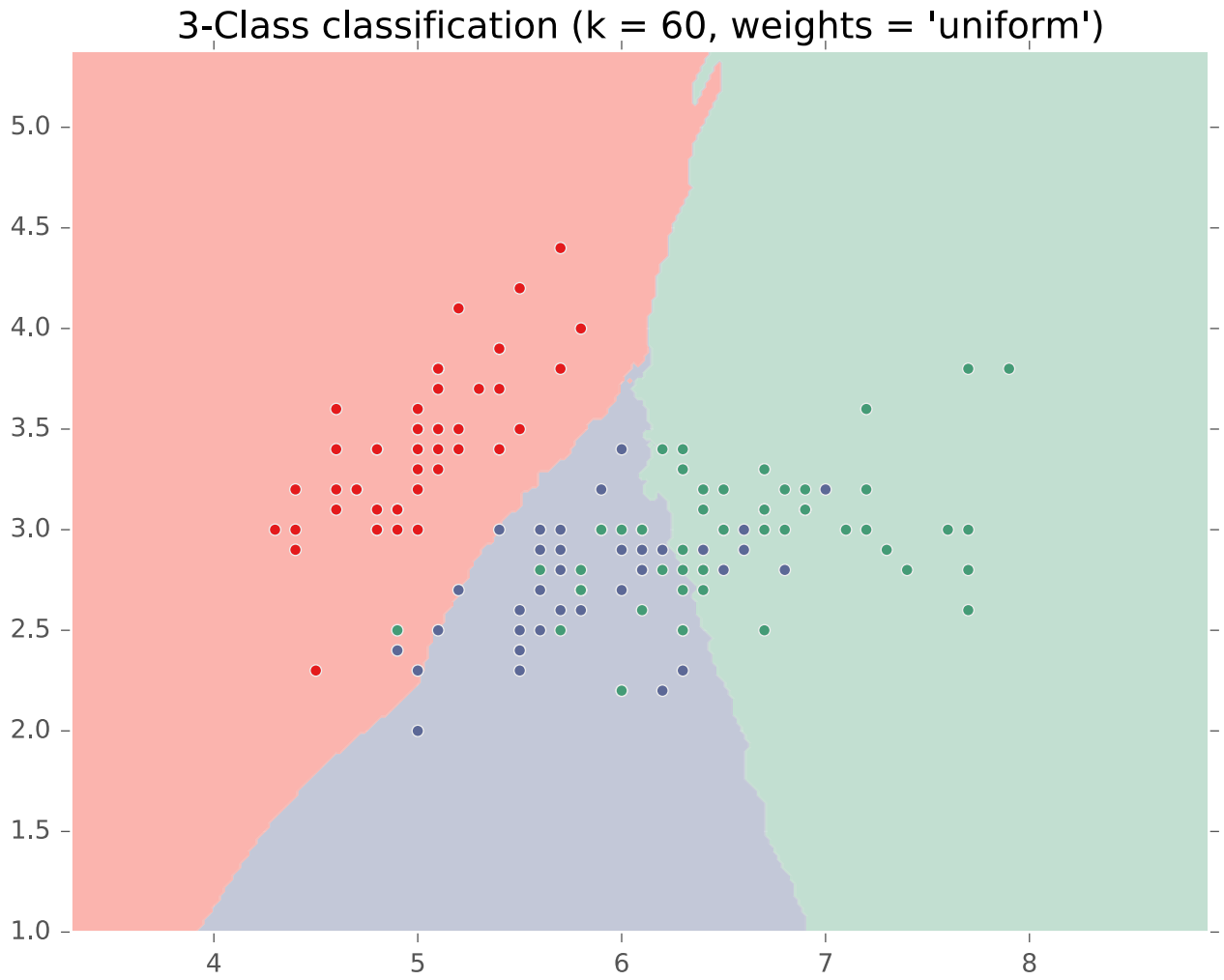
KNN on Fisher Iris Data



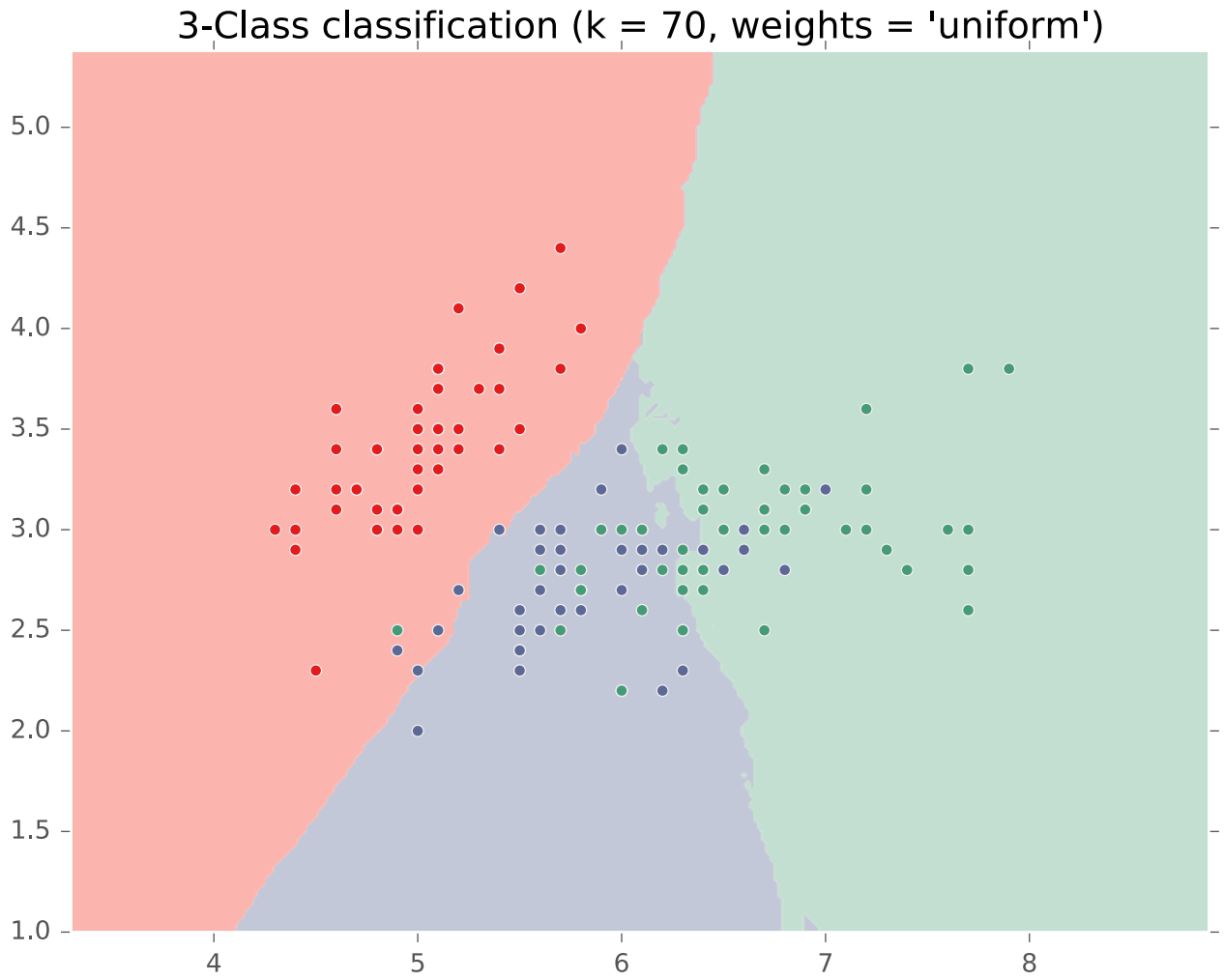
KNN on Fisher Iris Data



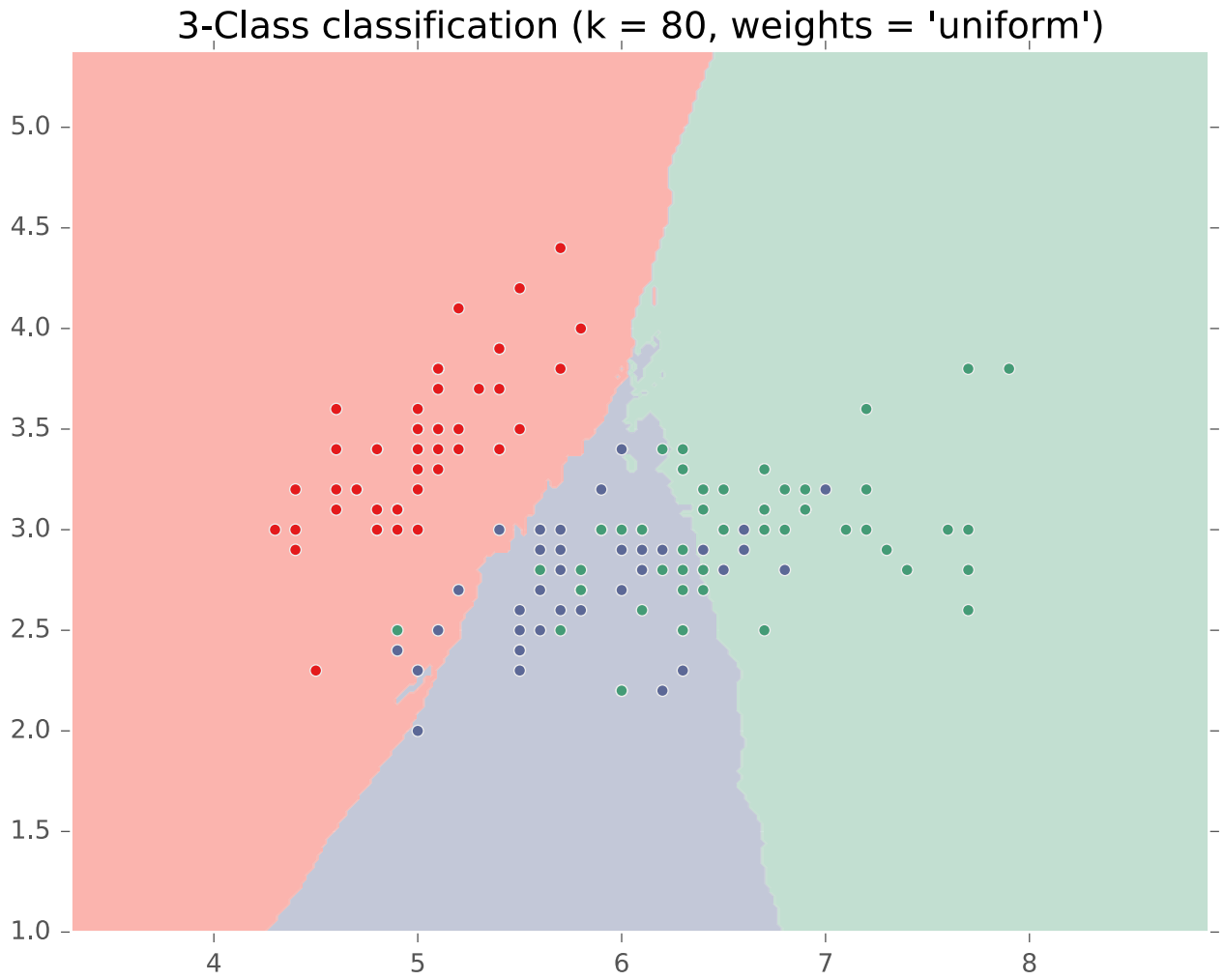
KNN on Fisher Iris Data



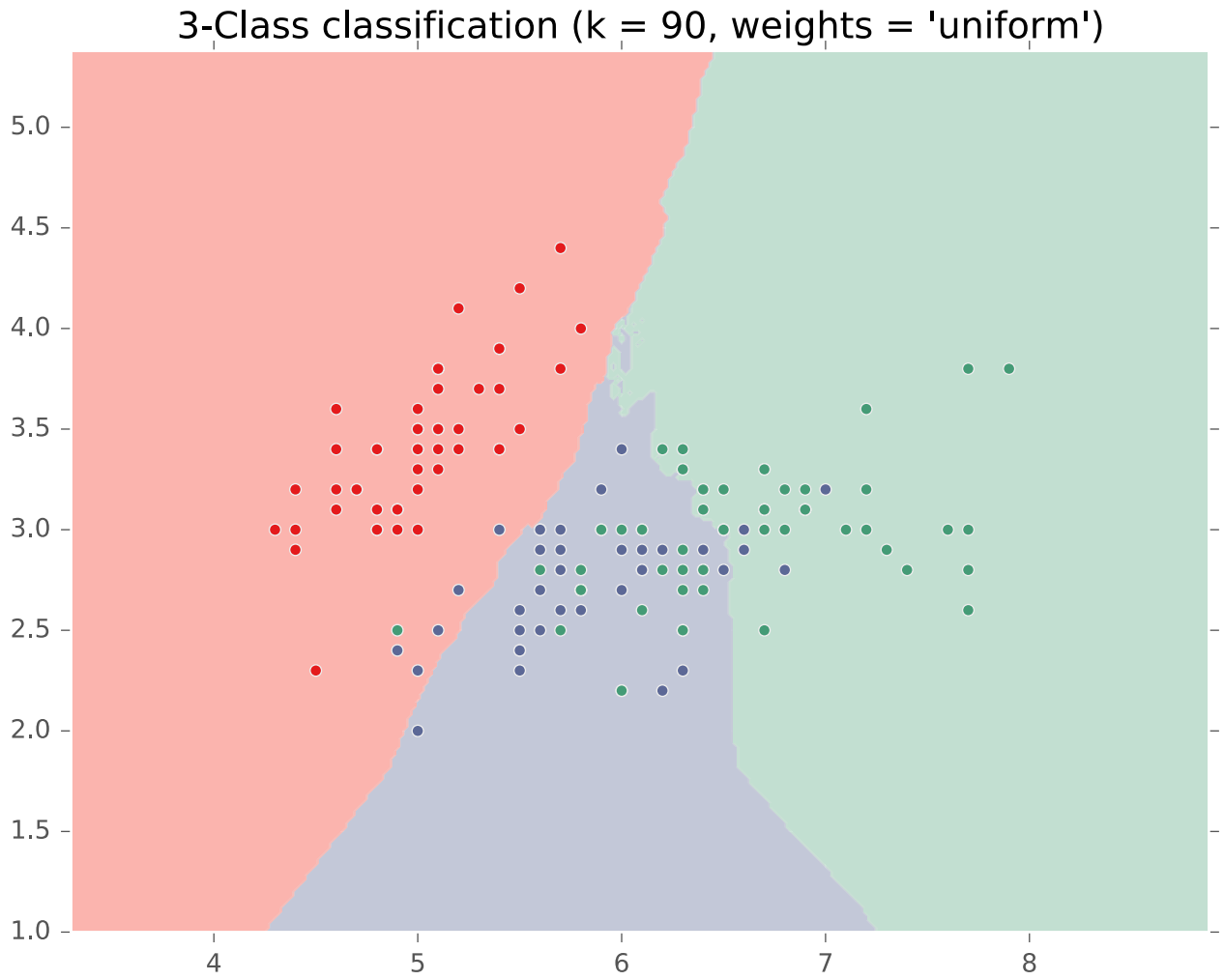
KNN on Fisher Iris Data



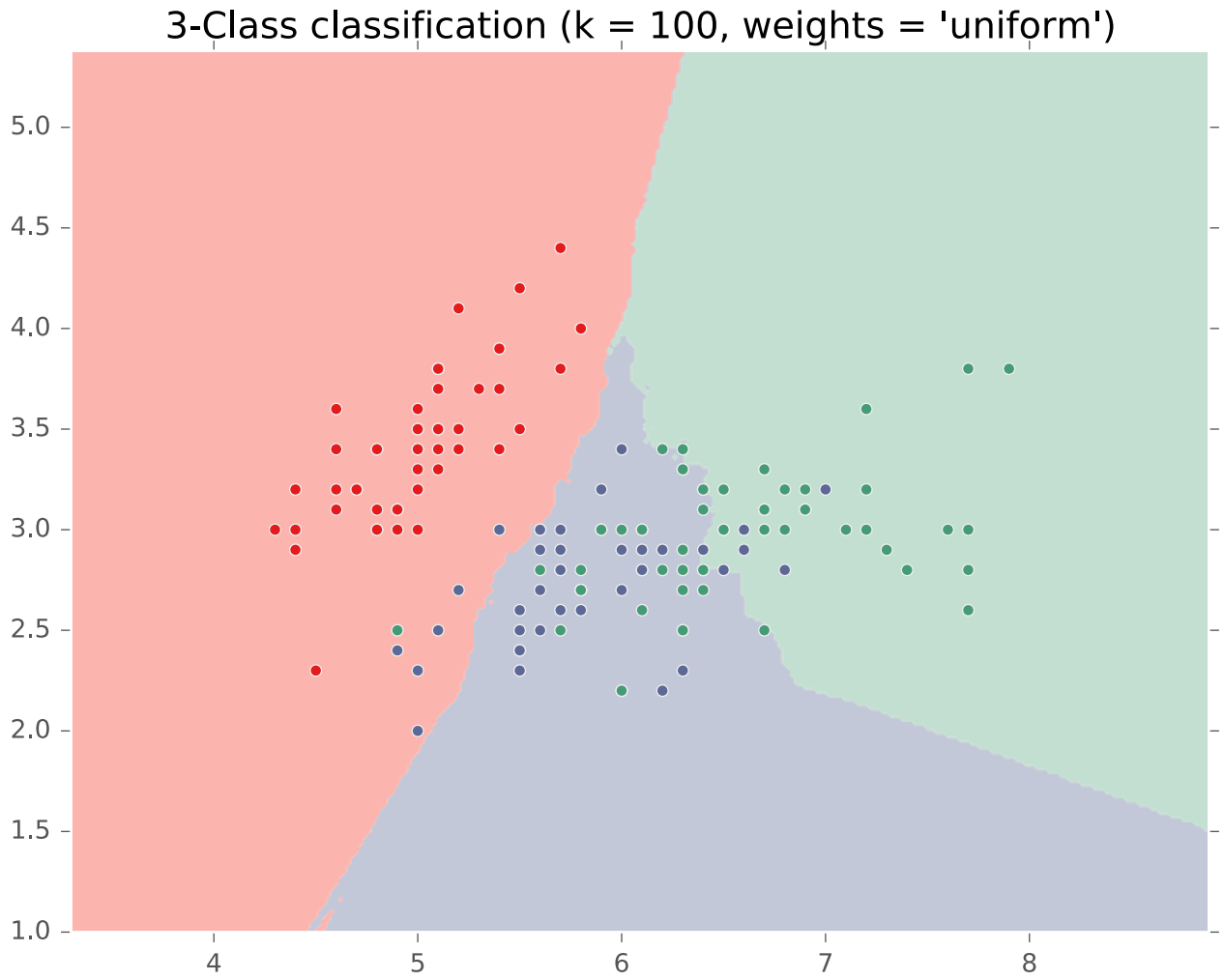
KNN on Fisher Iris Data



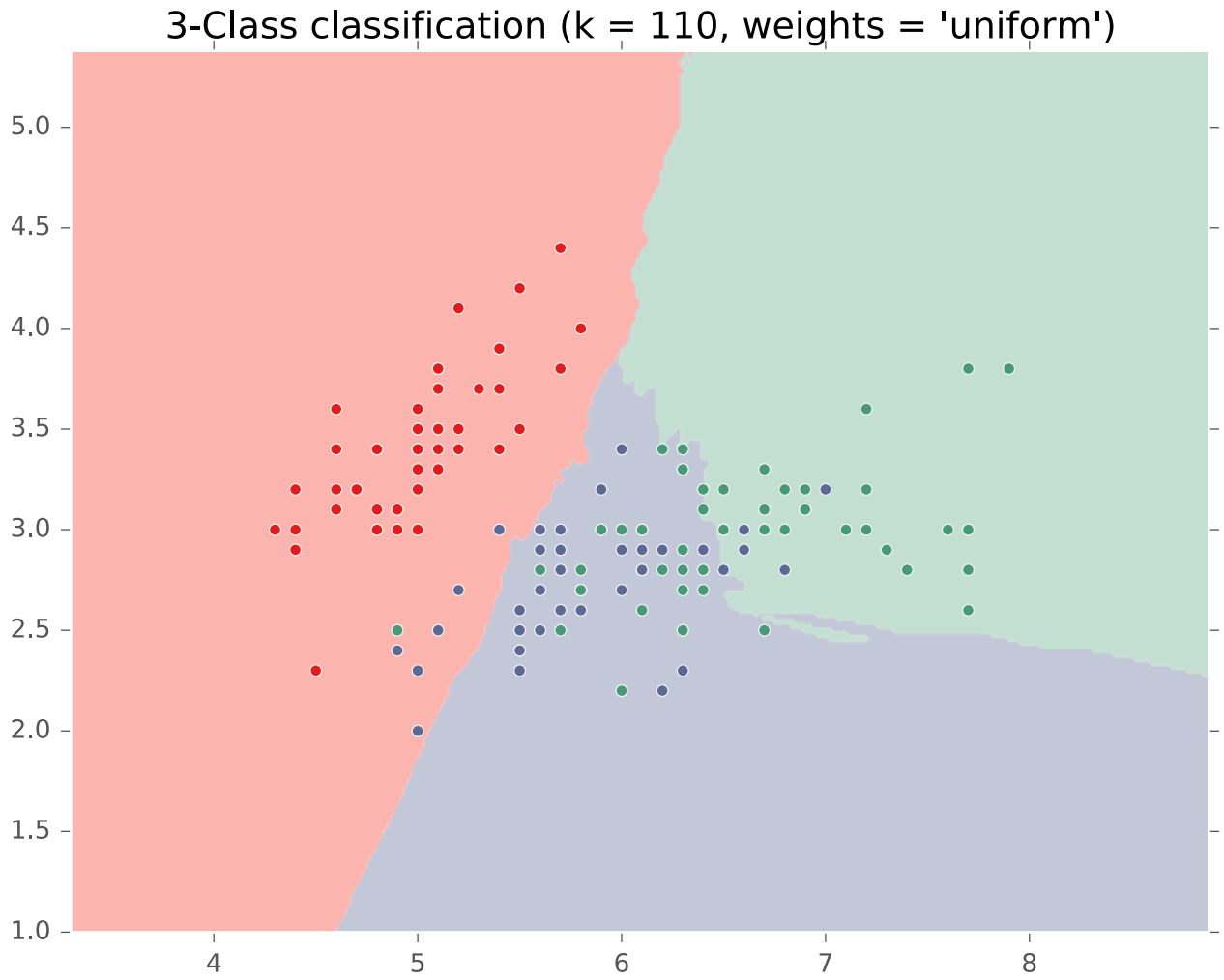
KNN on Fisher Iris Data



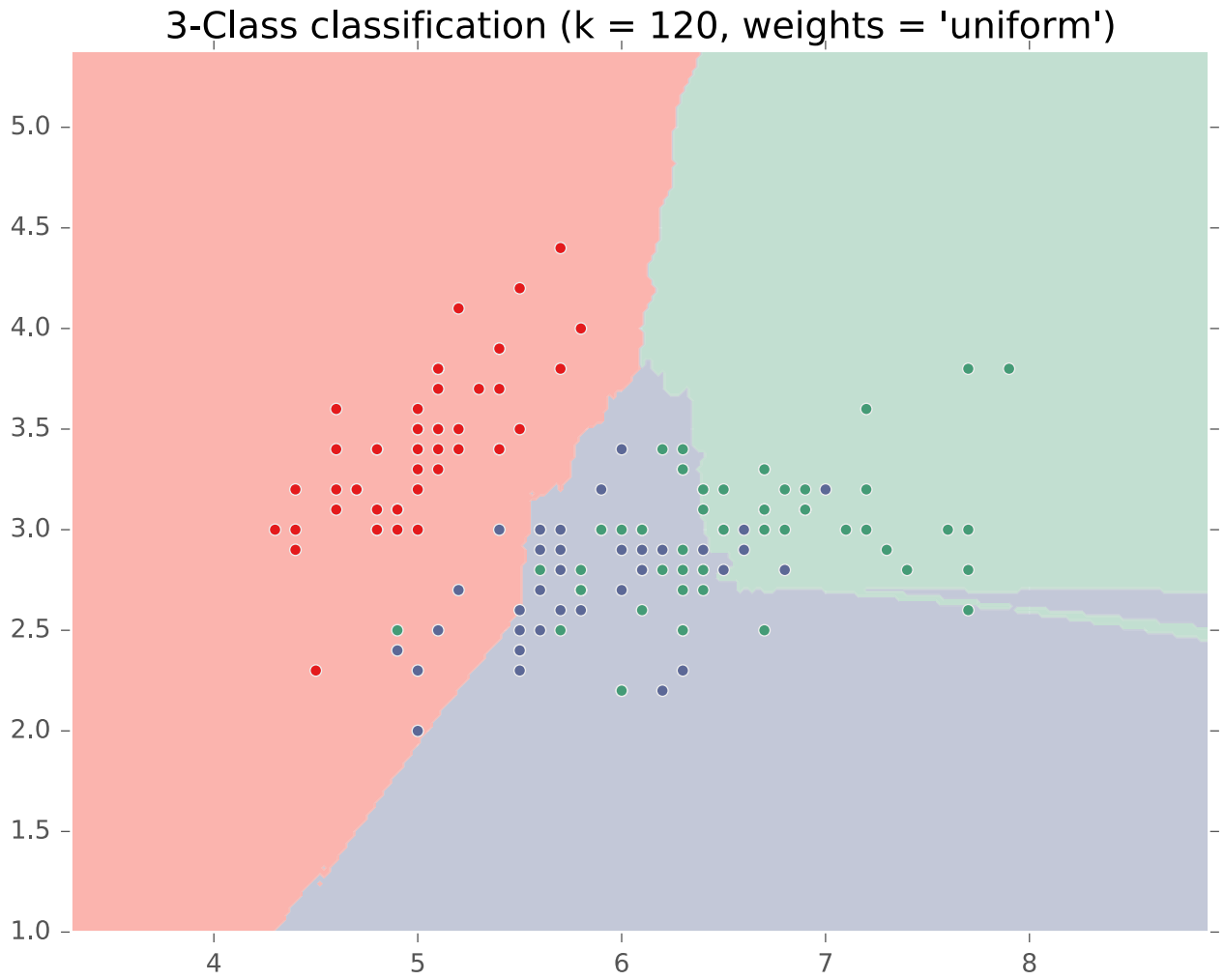
KNN on Fisher Iris Data



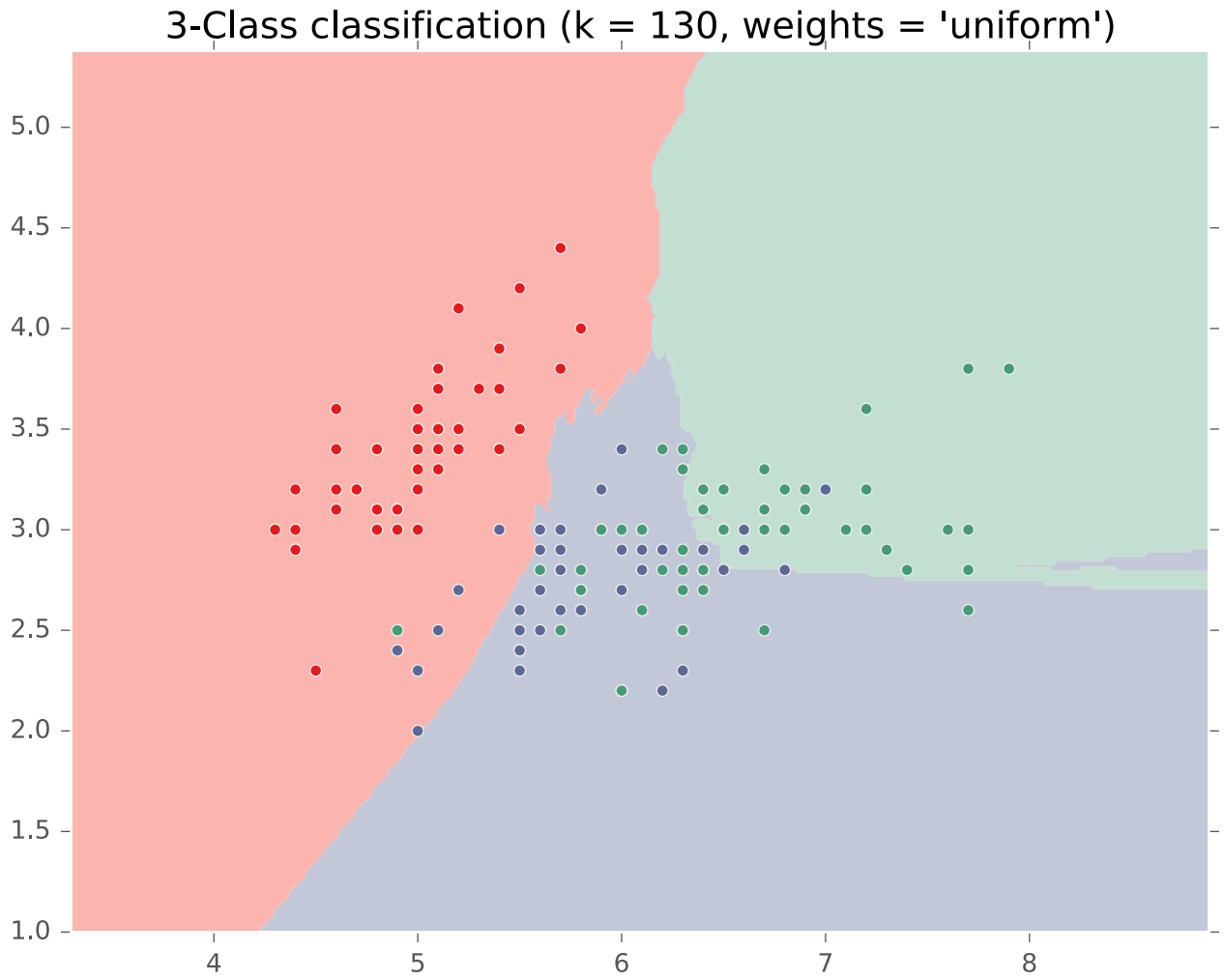
KNN on Fisher Iris Data



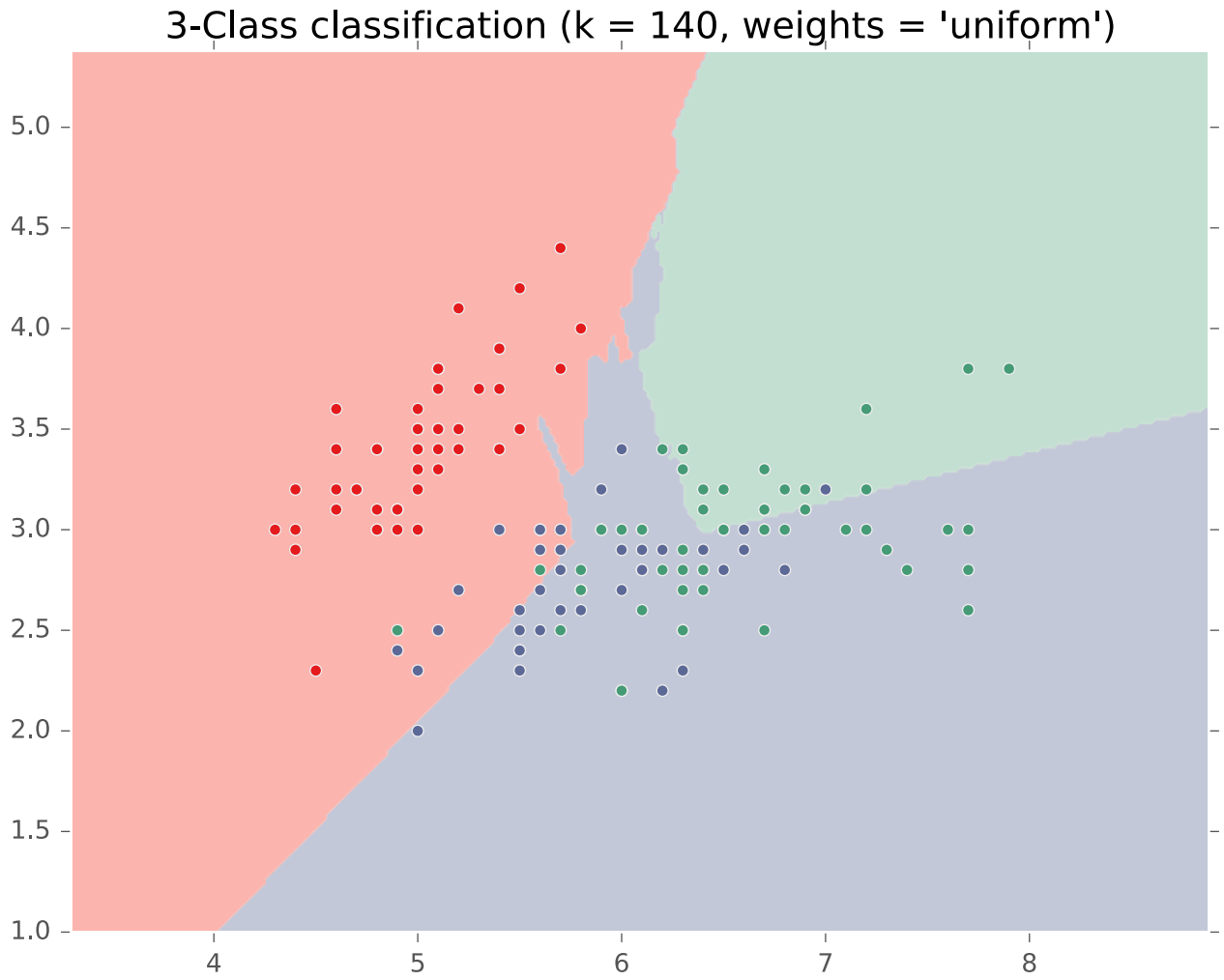
KNN on Fisher Iris Data



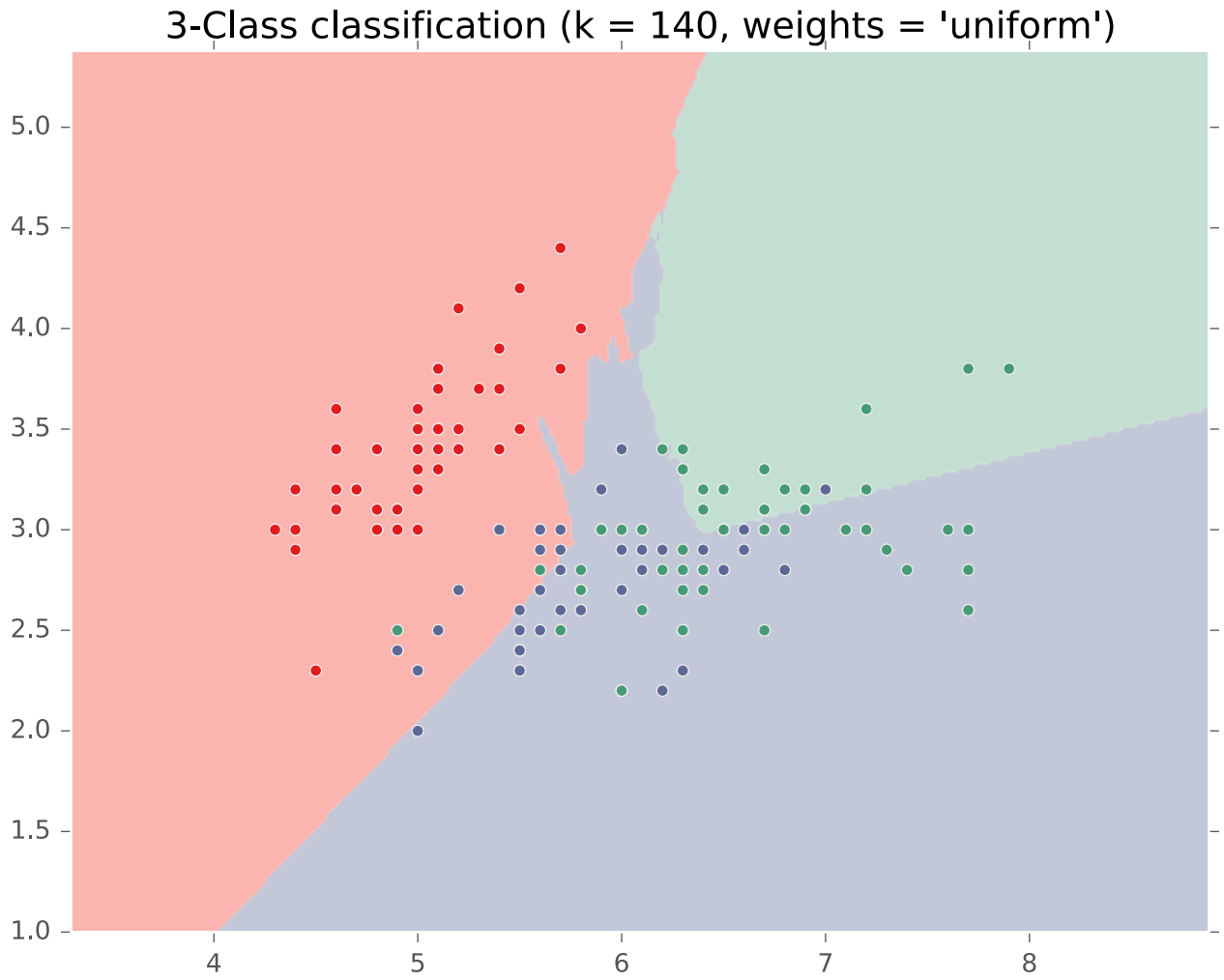
KNN on Fisher Iris Data



KNN on Fisher Iris Data

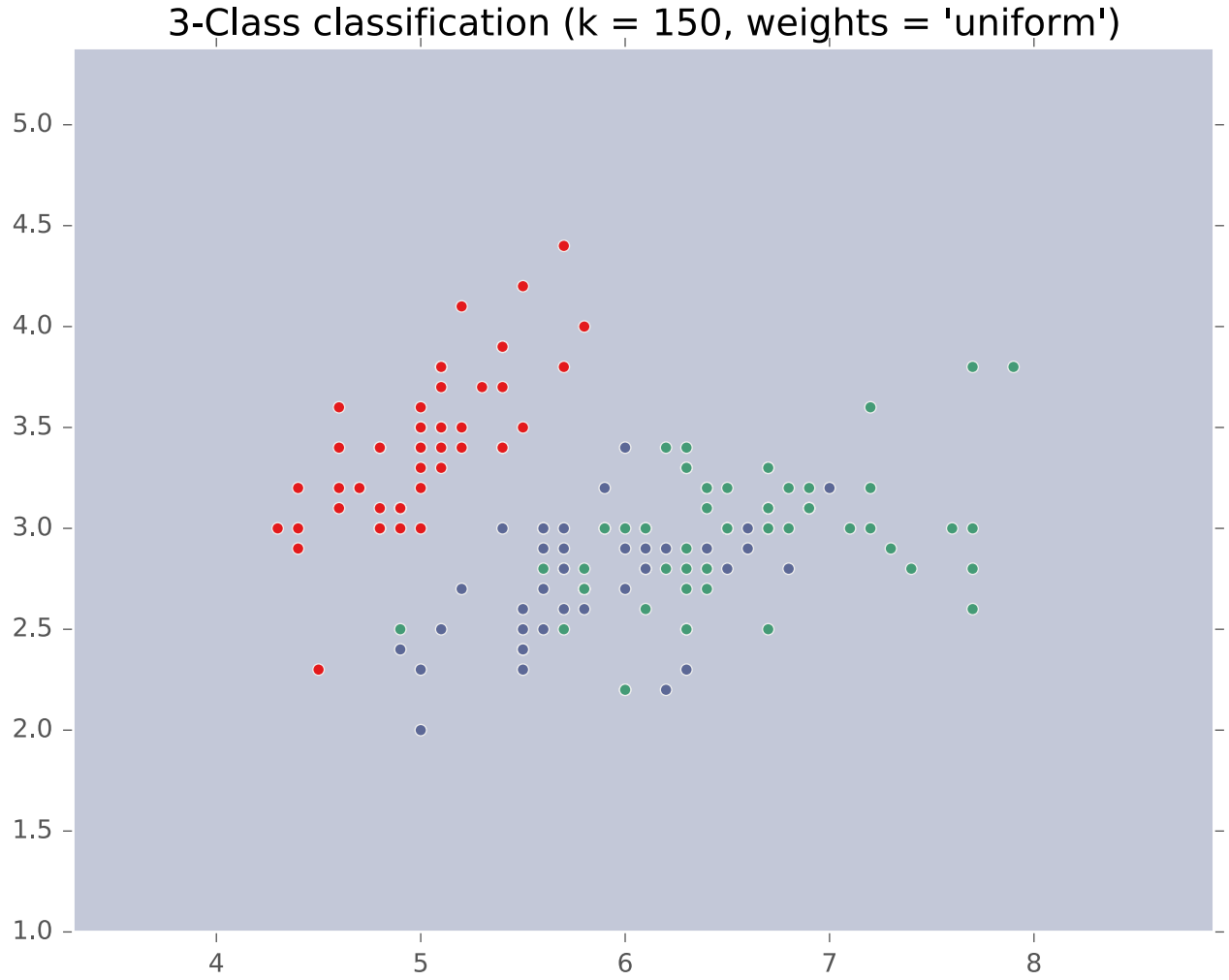


KNN on Fisher Iris Data



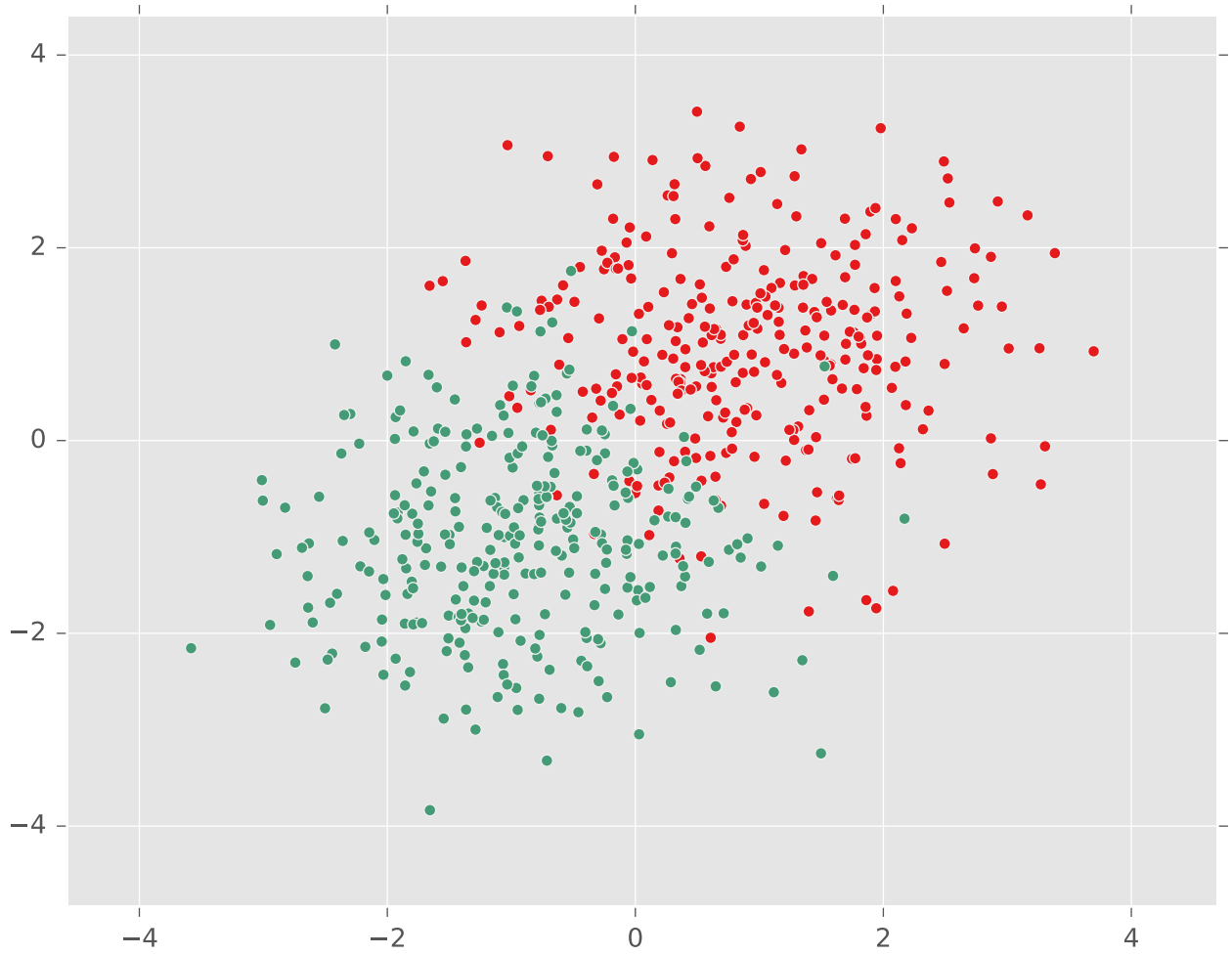
KNN on Fisher Iris Data

Special Case: Majority Vote

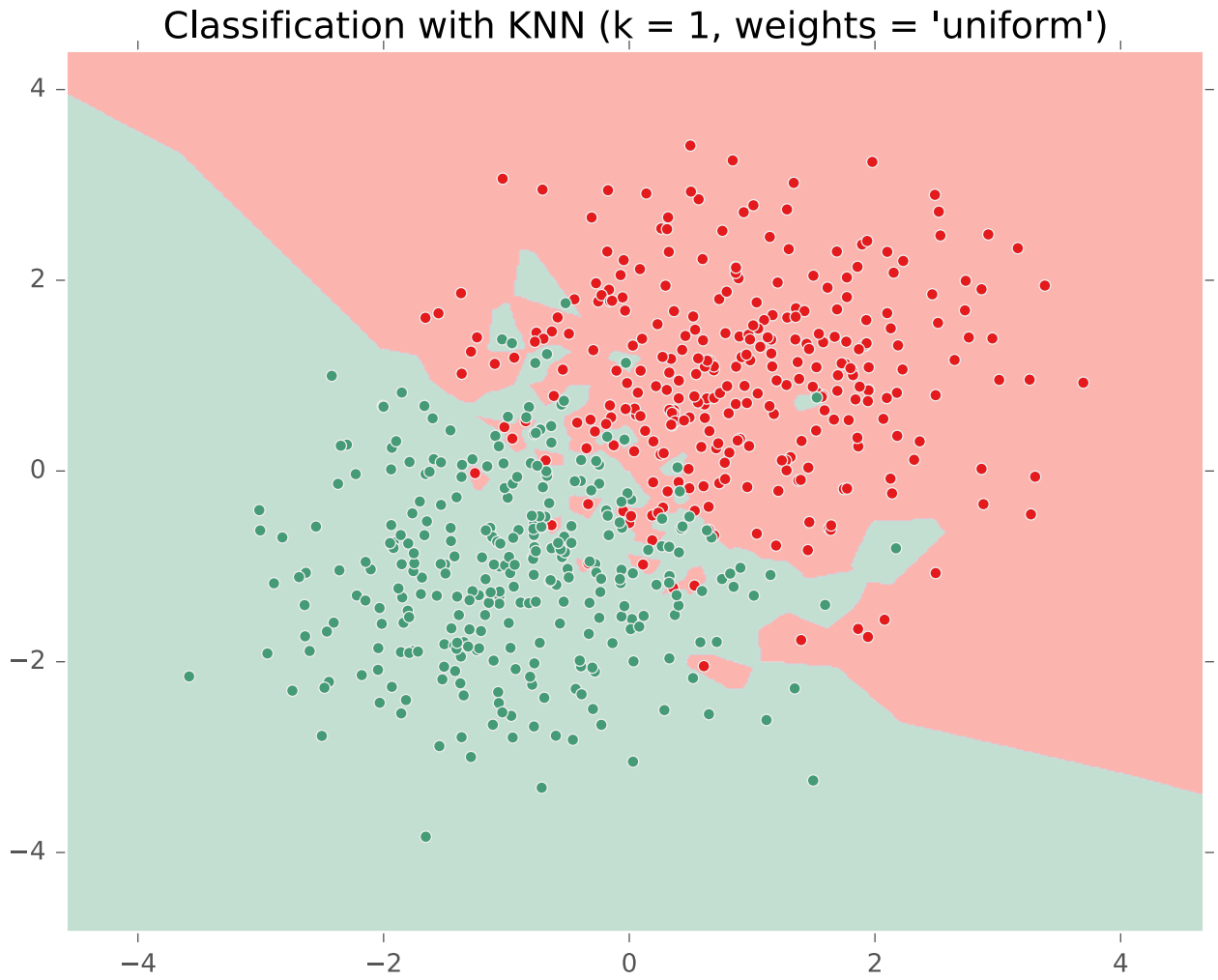


KNN ON GAUSSIAN DATA

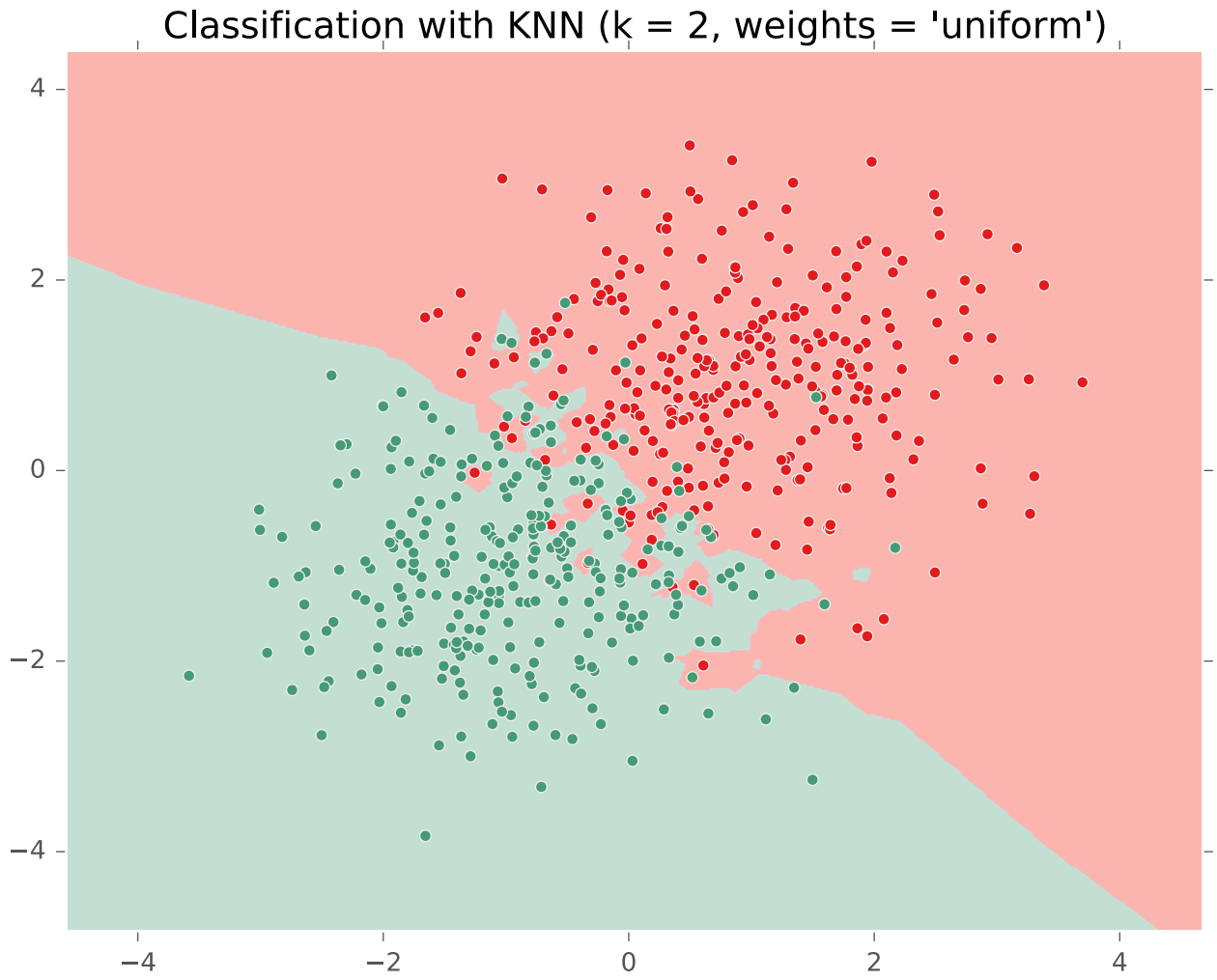
KNN on Gaussian Data



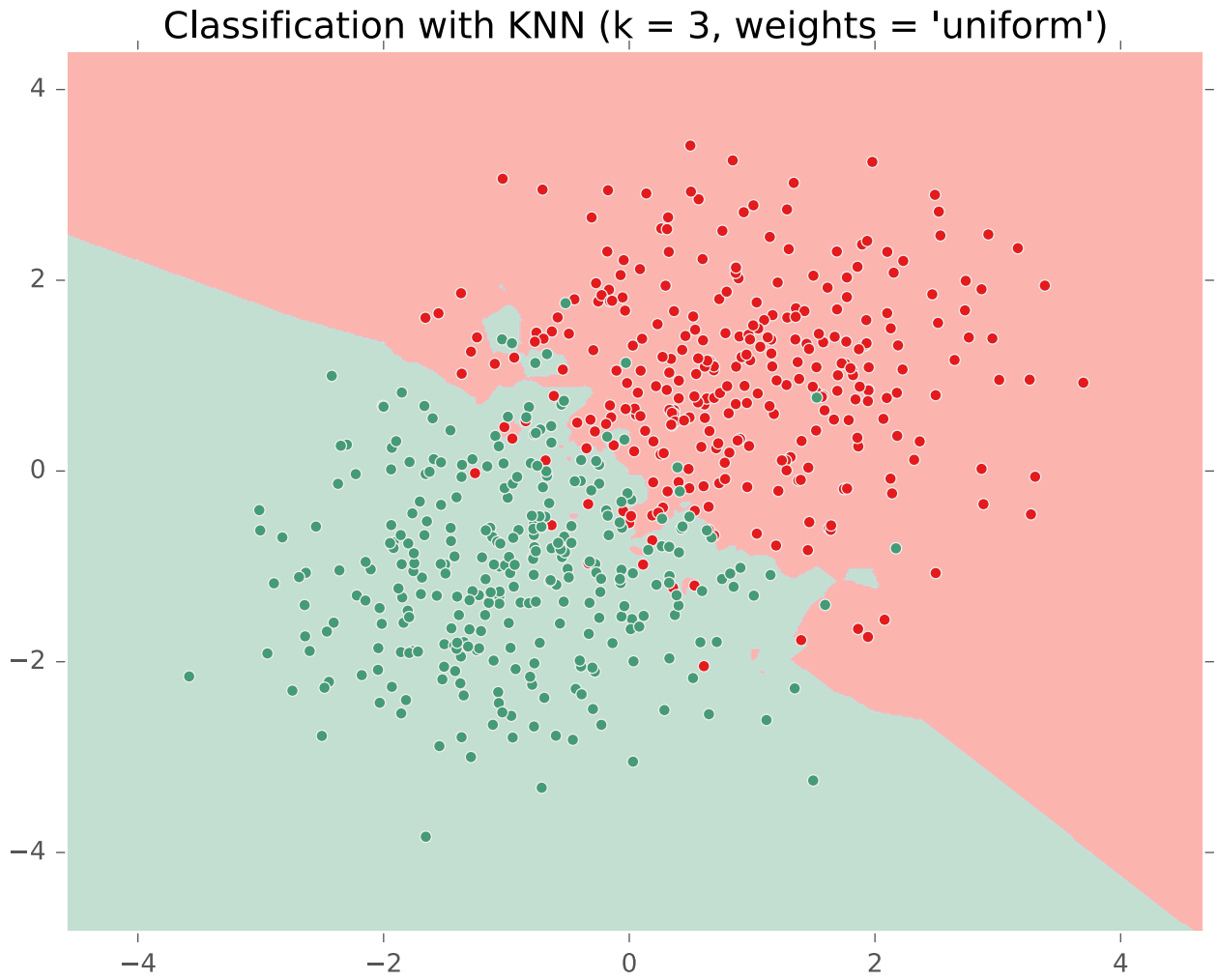
KNN on Gaussian Data



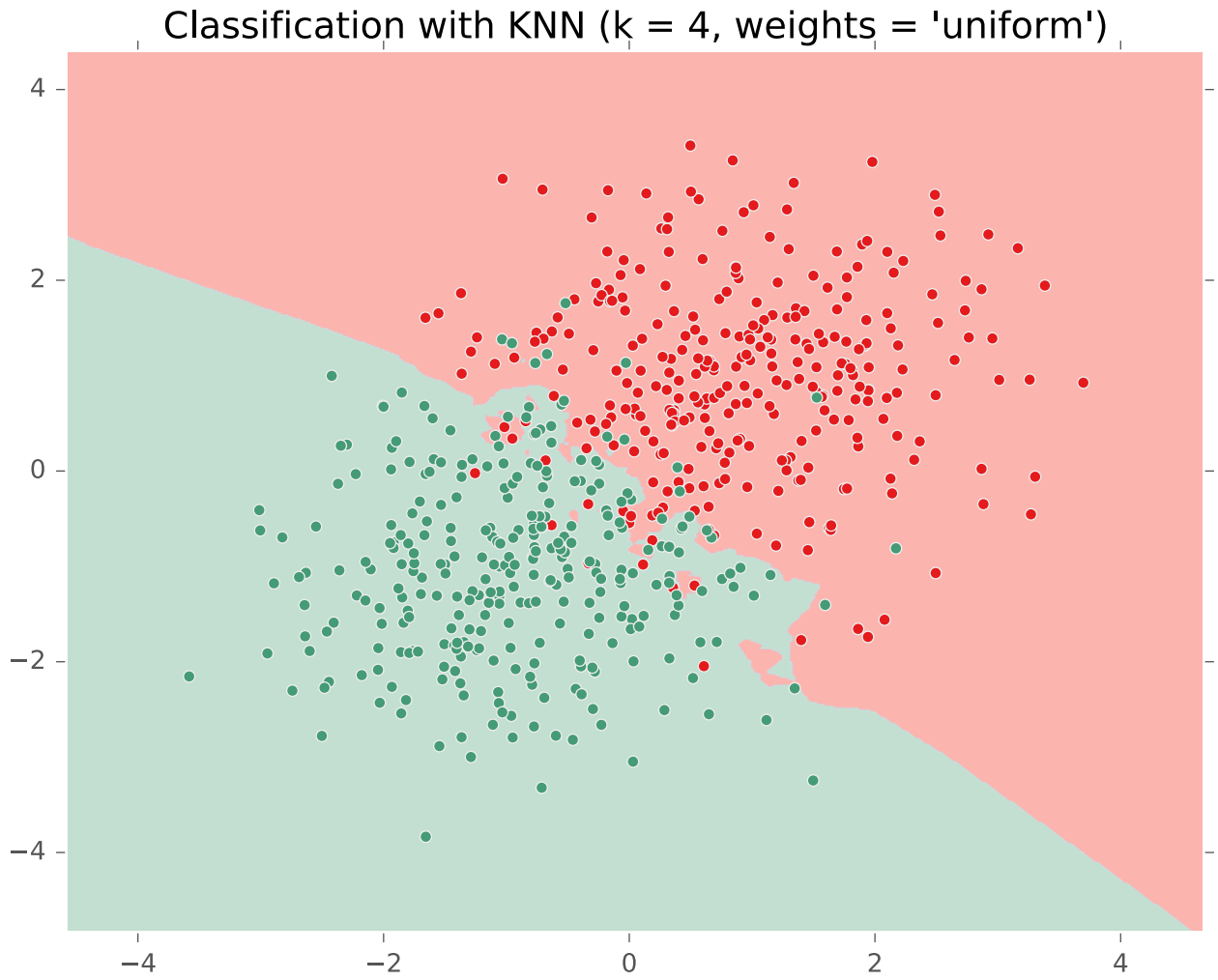
KNN on Gaussian Data



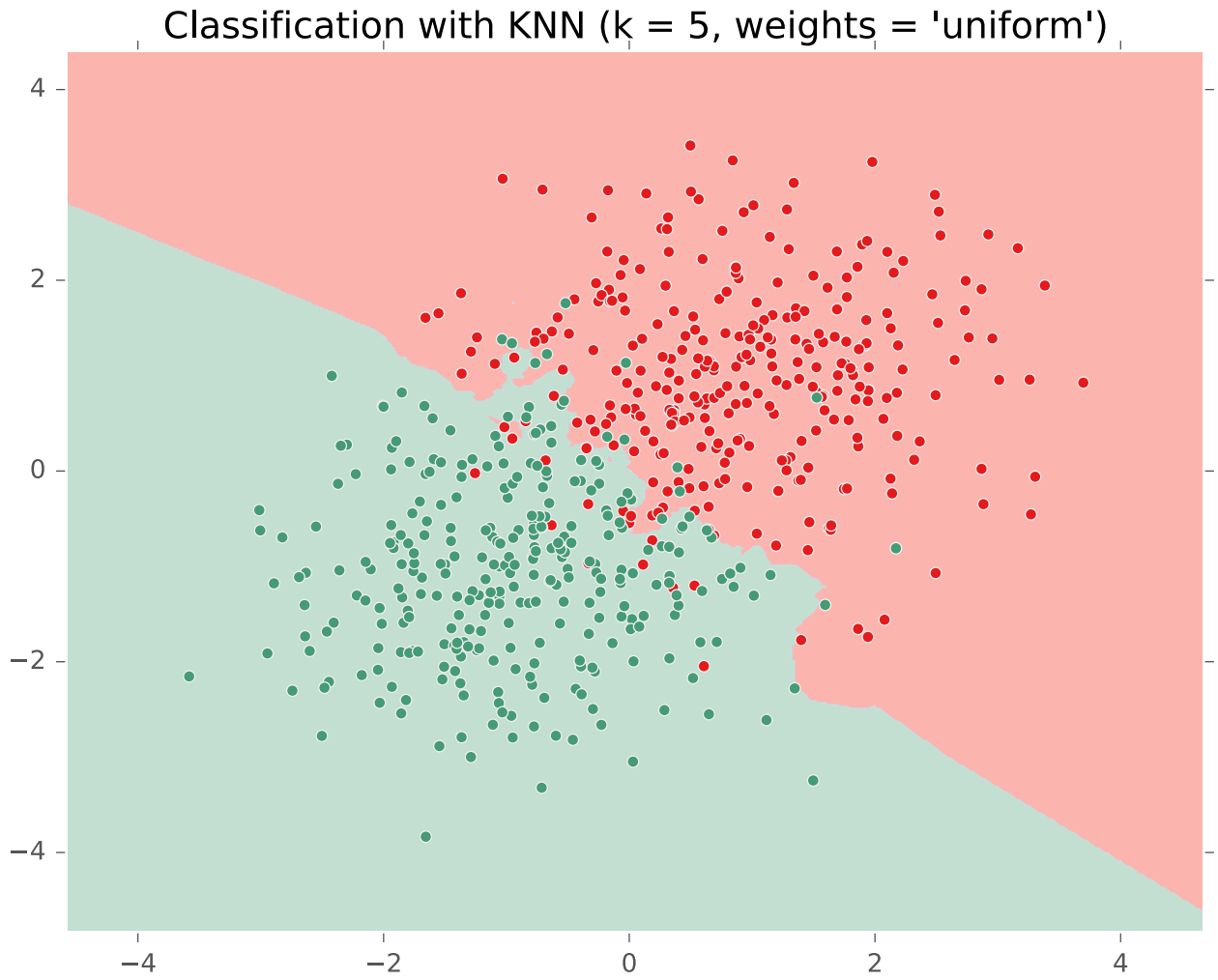
KNN on Gaussian Data



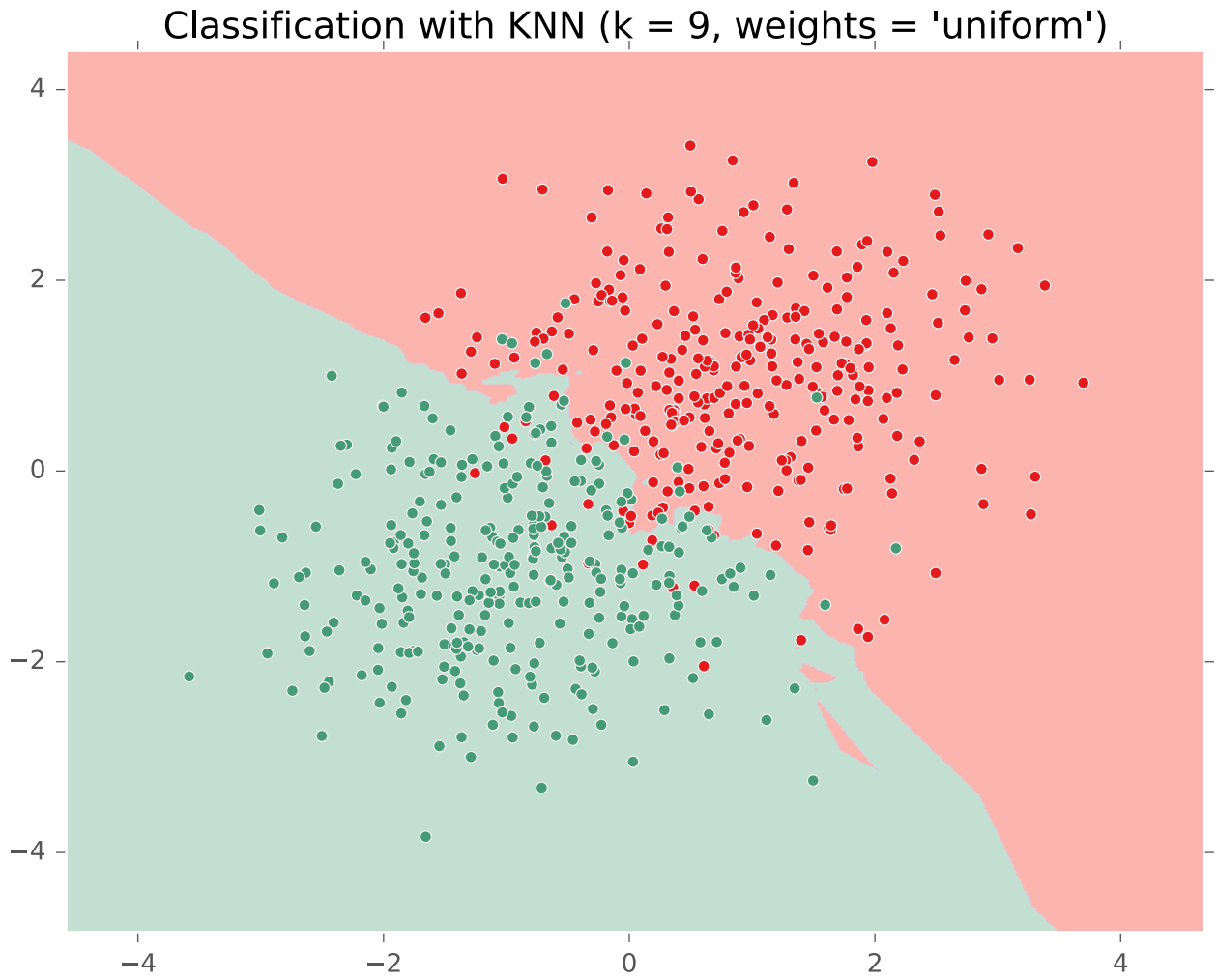
KNN on Gaussian Data



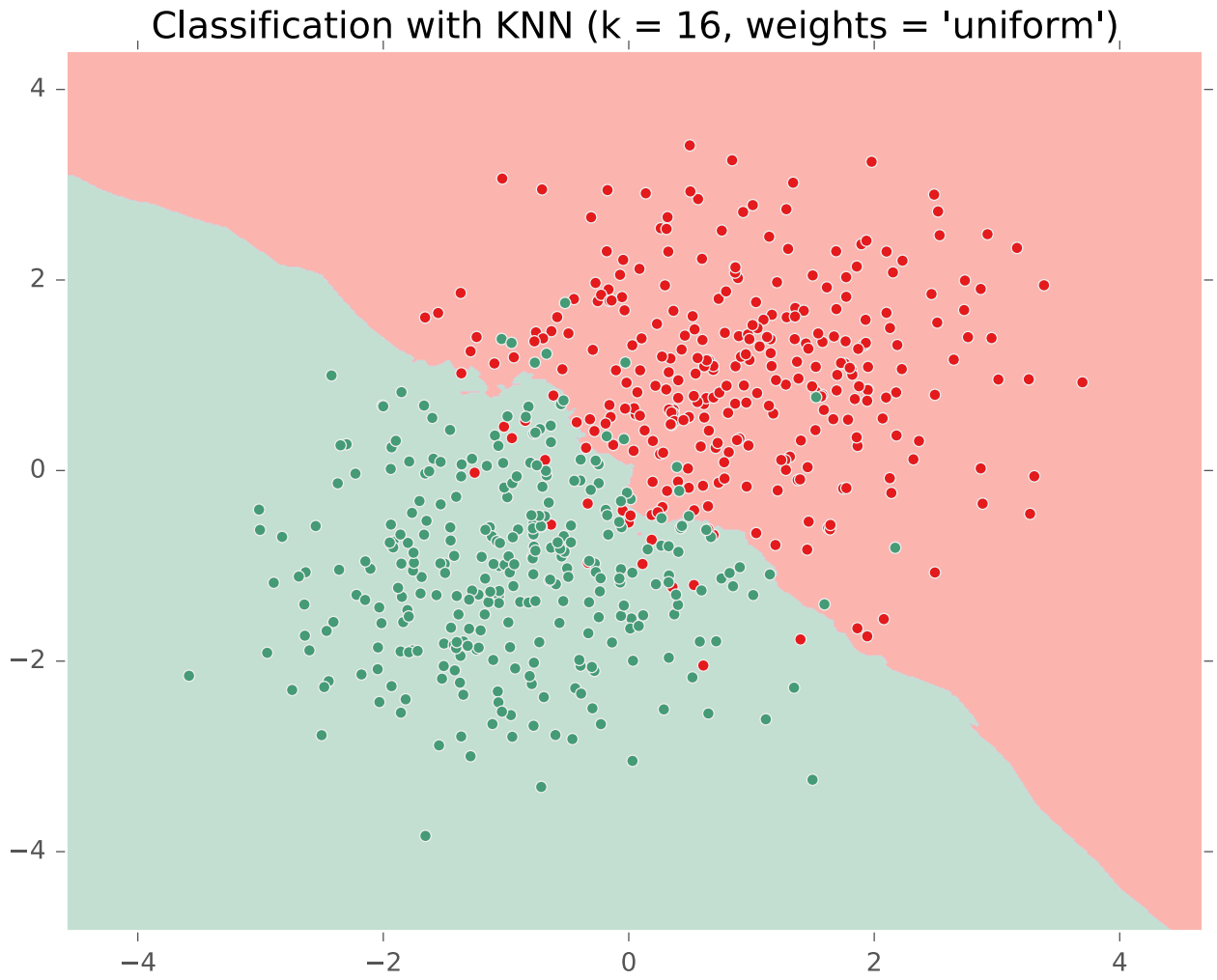
KNN on Gaussian Data



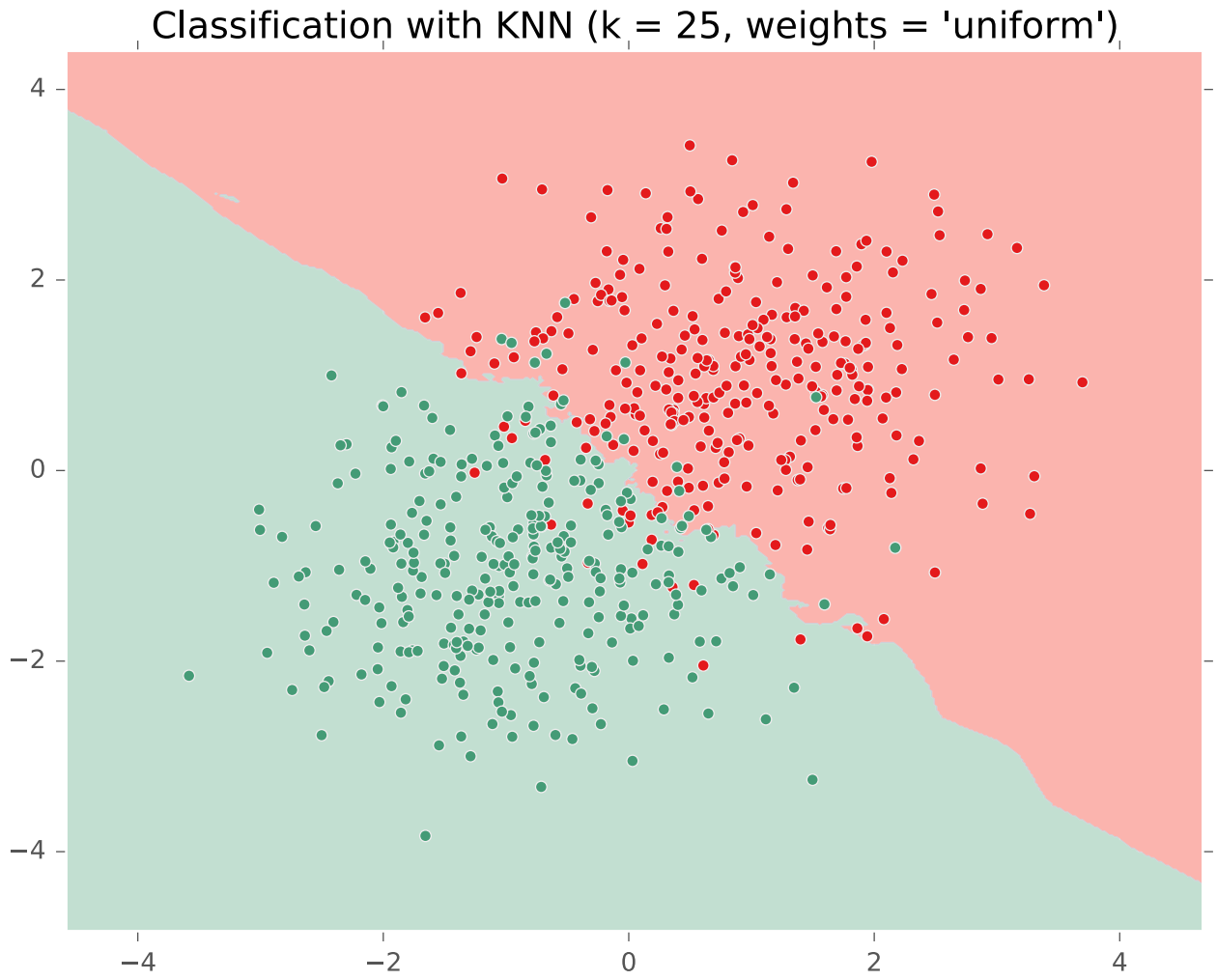
KNN on Gaussian Data



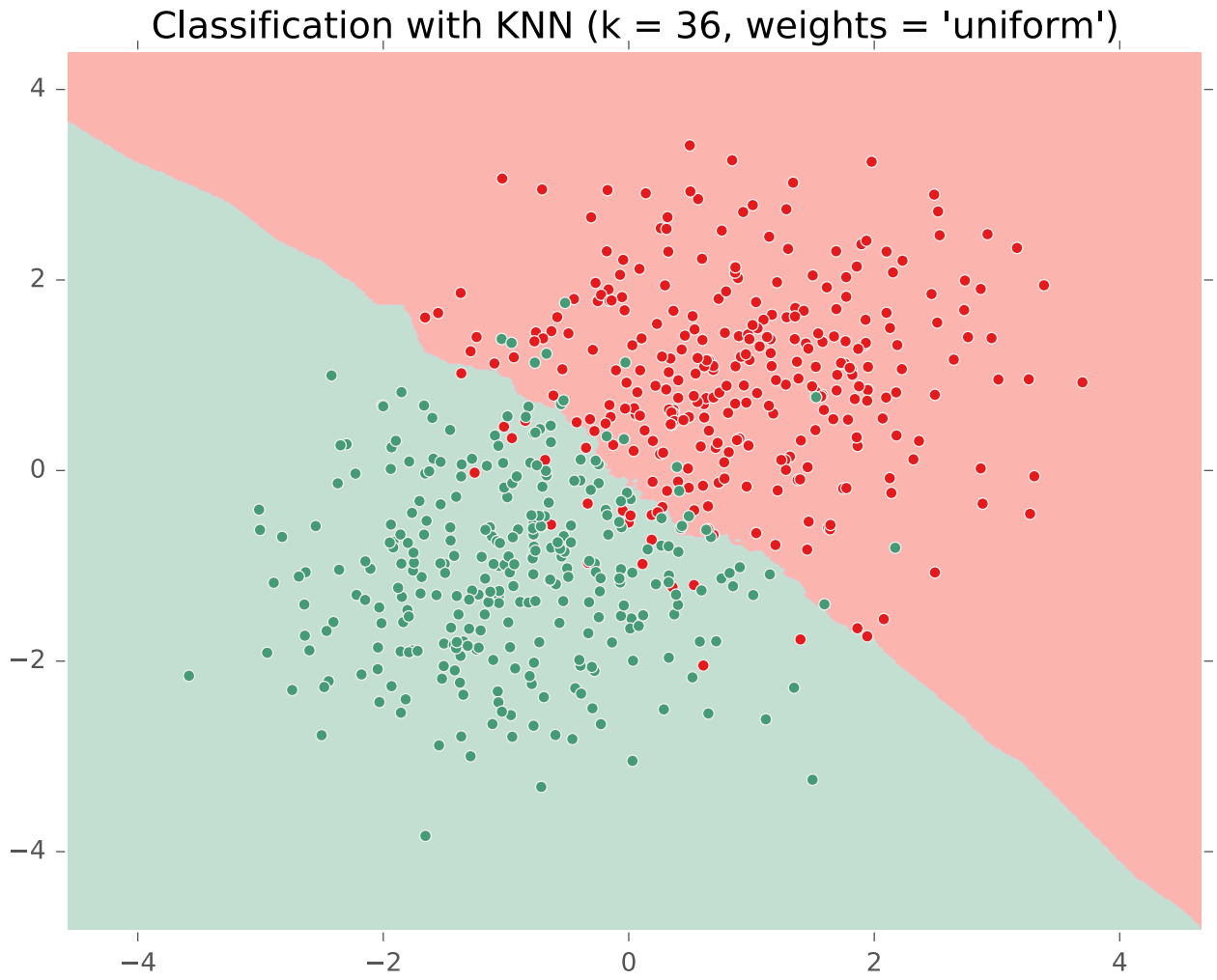
KNN on Gaussian Data



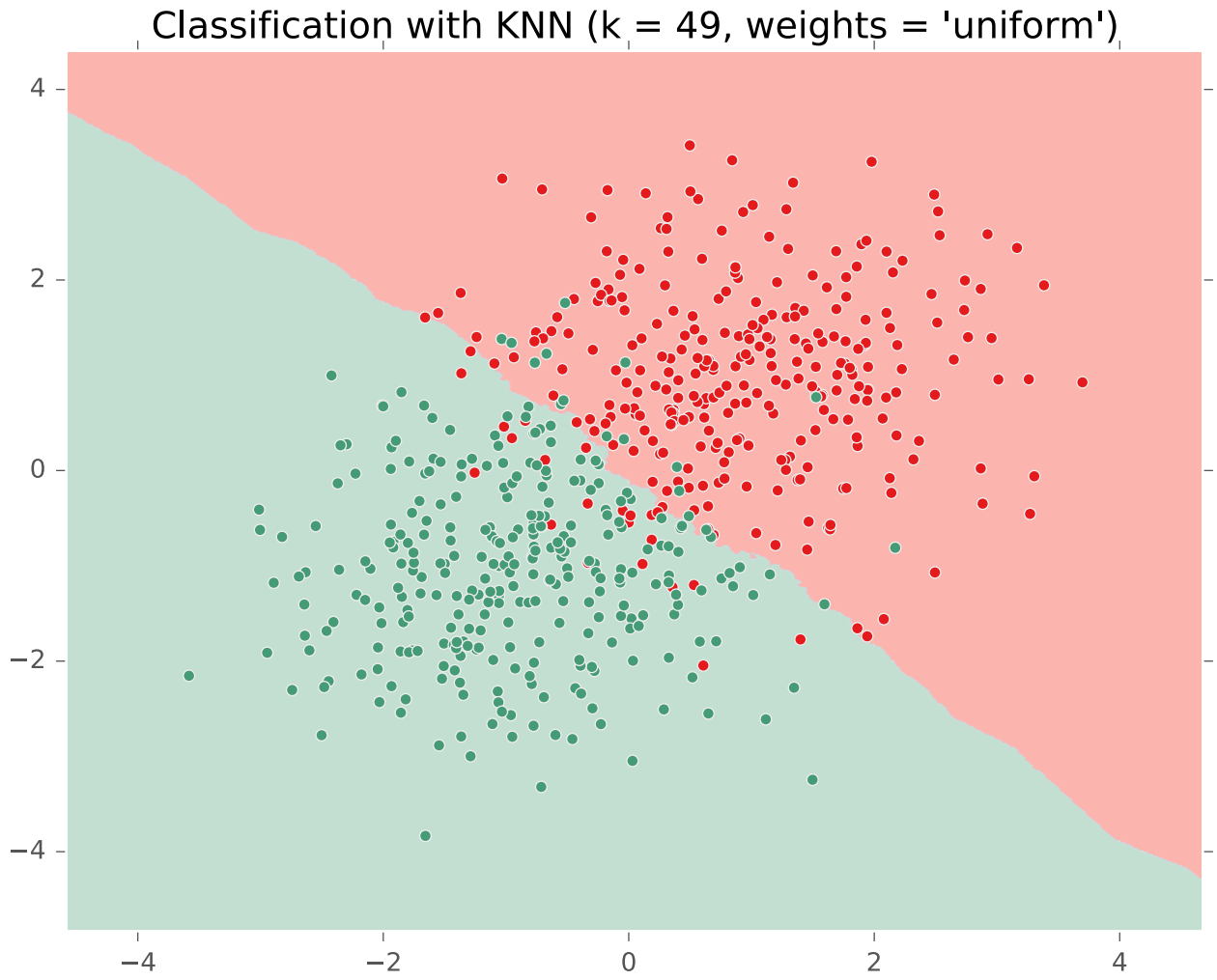
KNN on Gaussian Data



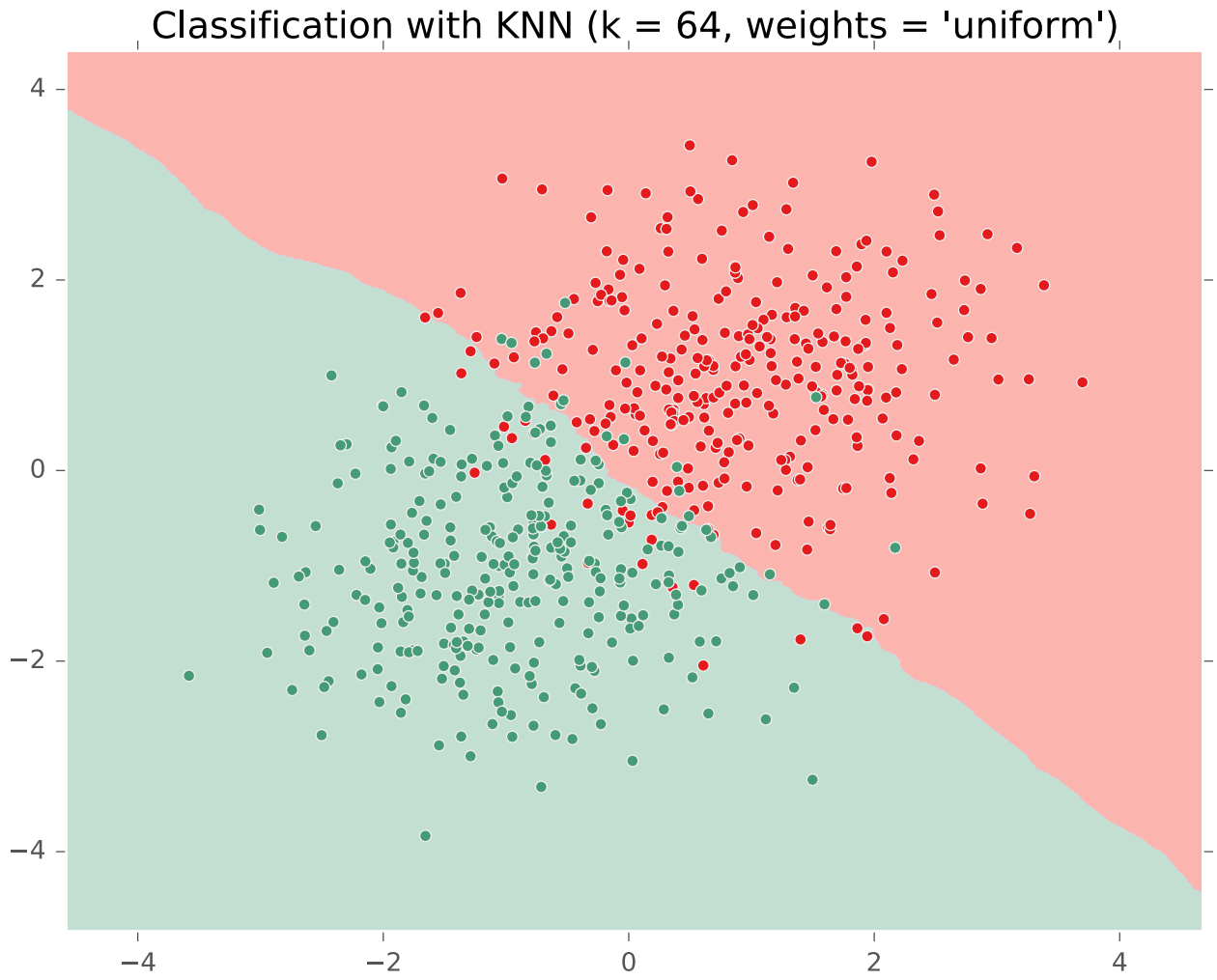
KNN on Gaussian Data



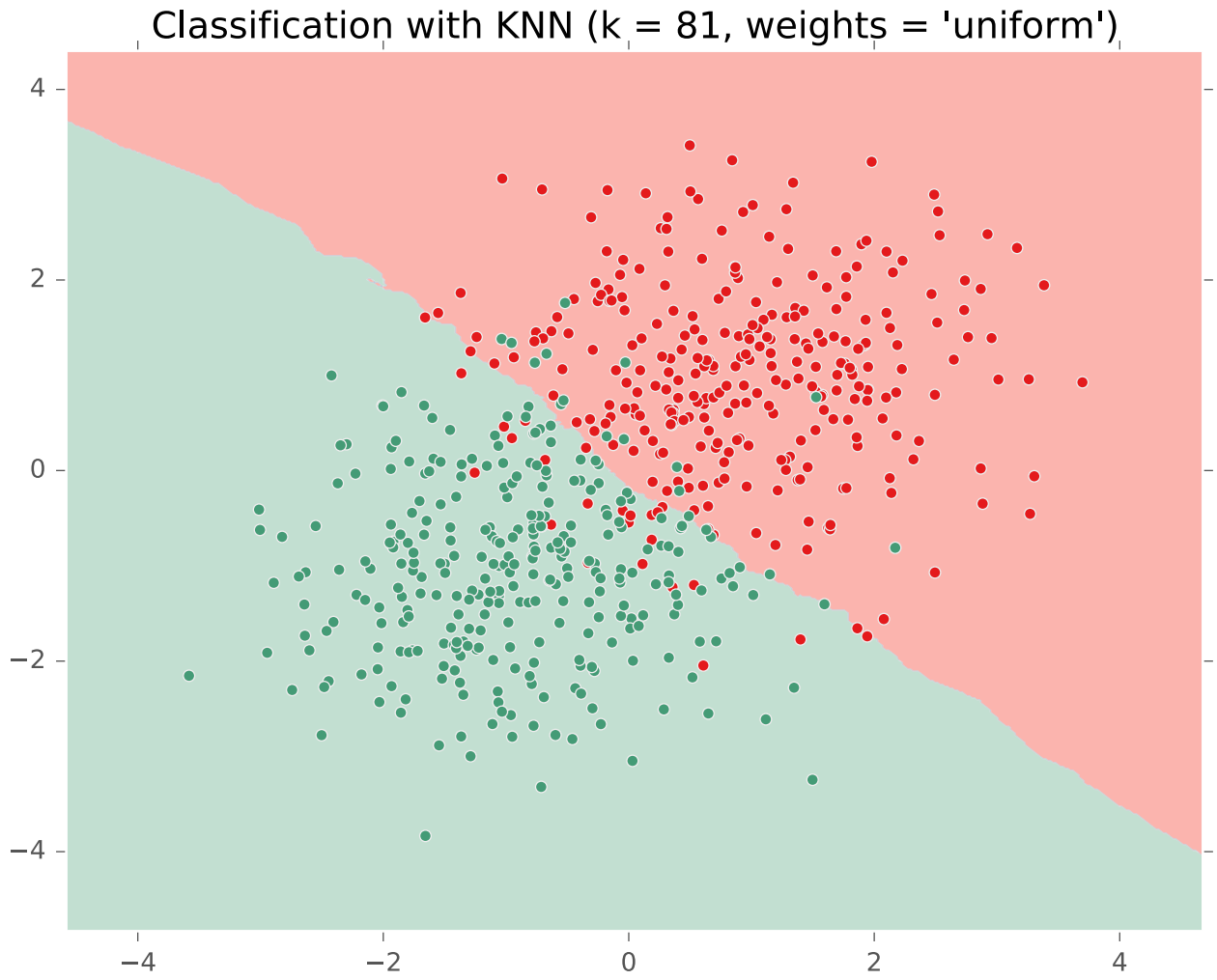
KNN on Gaussian Data



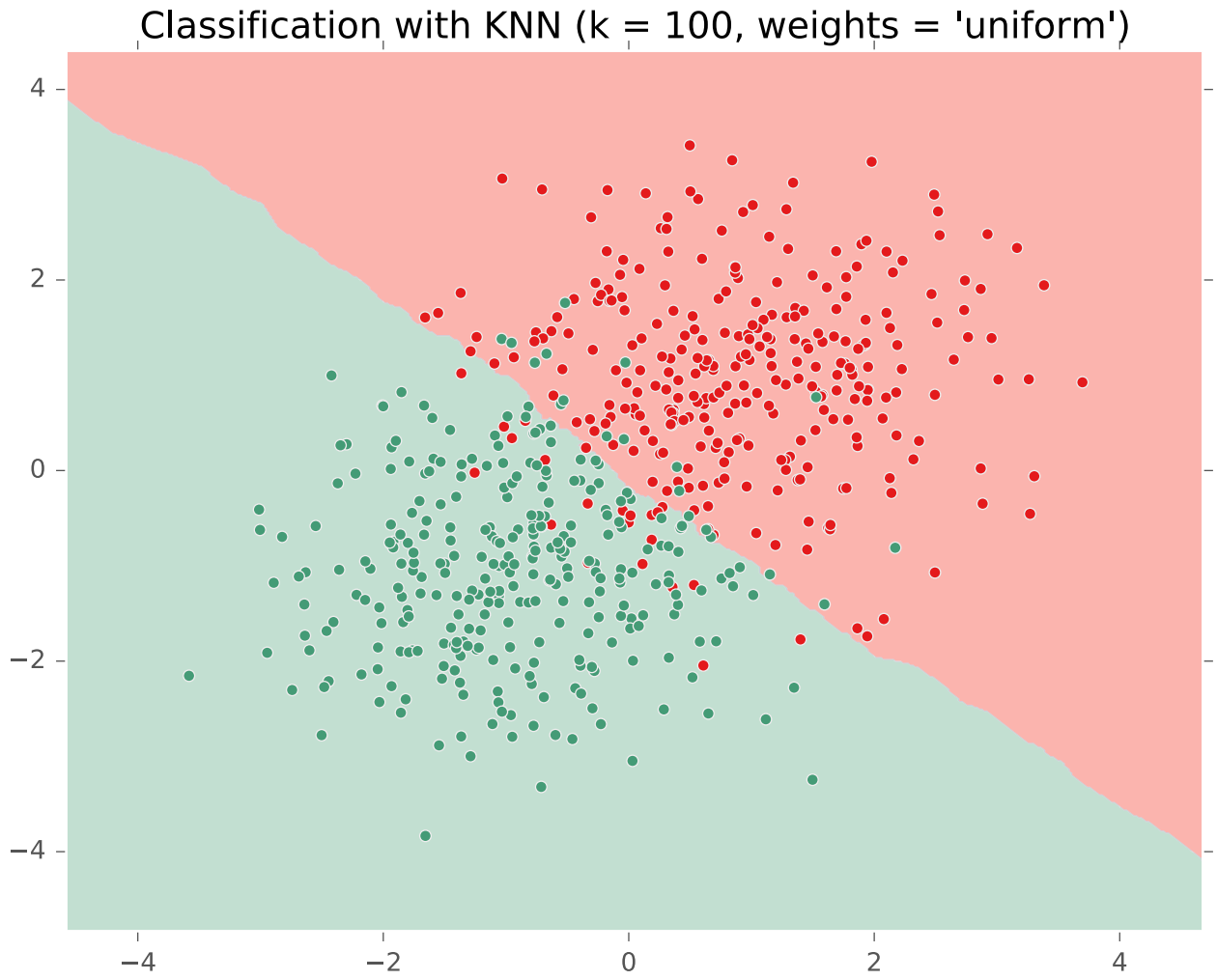
KNN on Gaussian Data



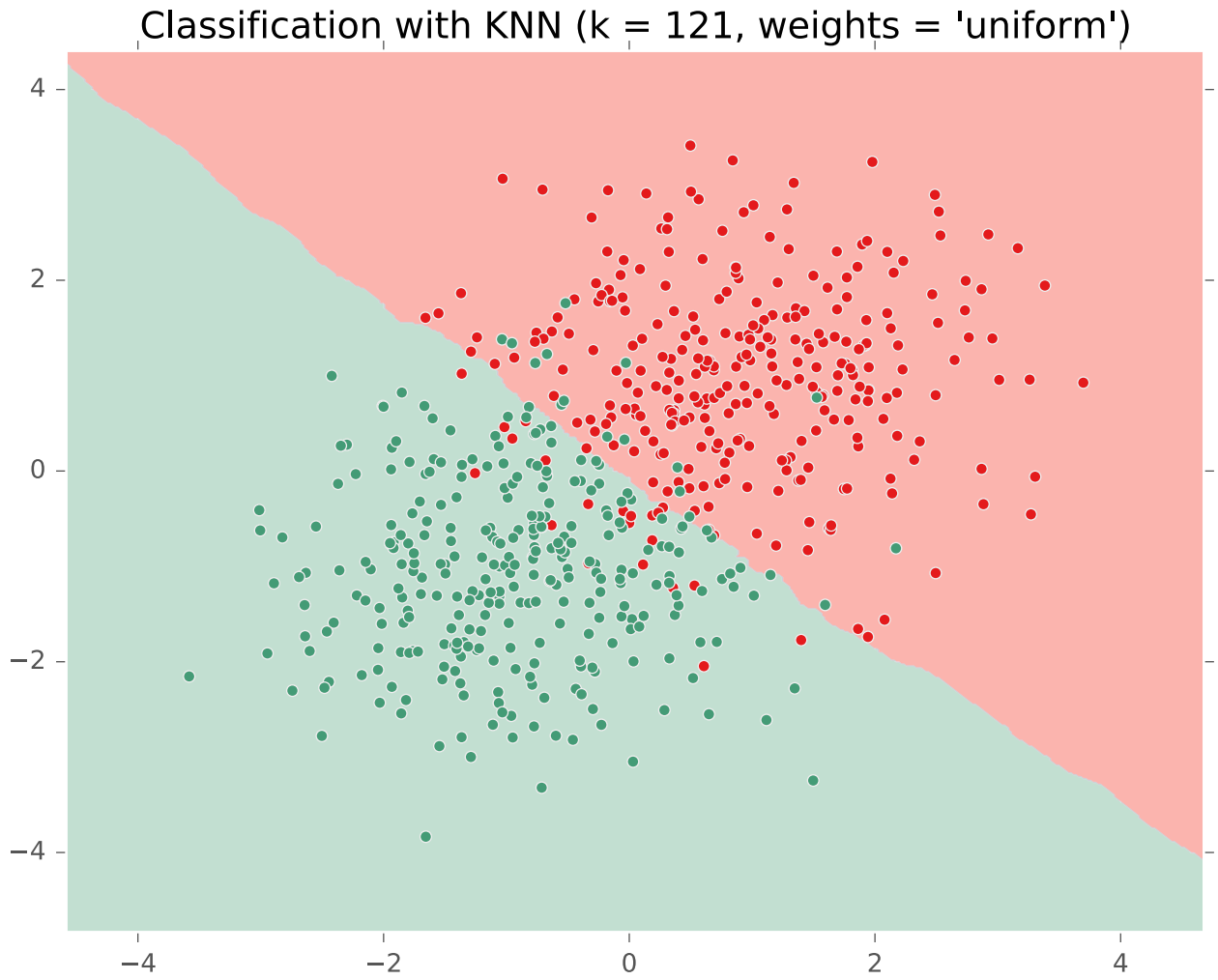
KNN on Gaussian Data



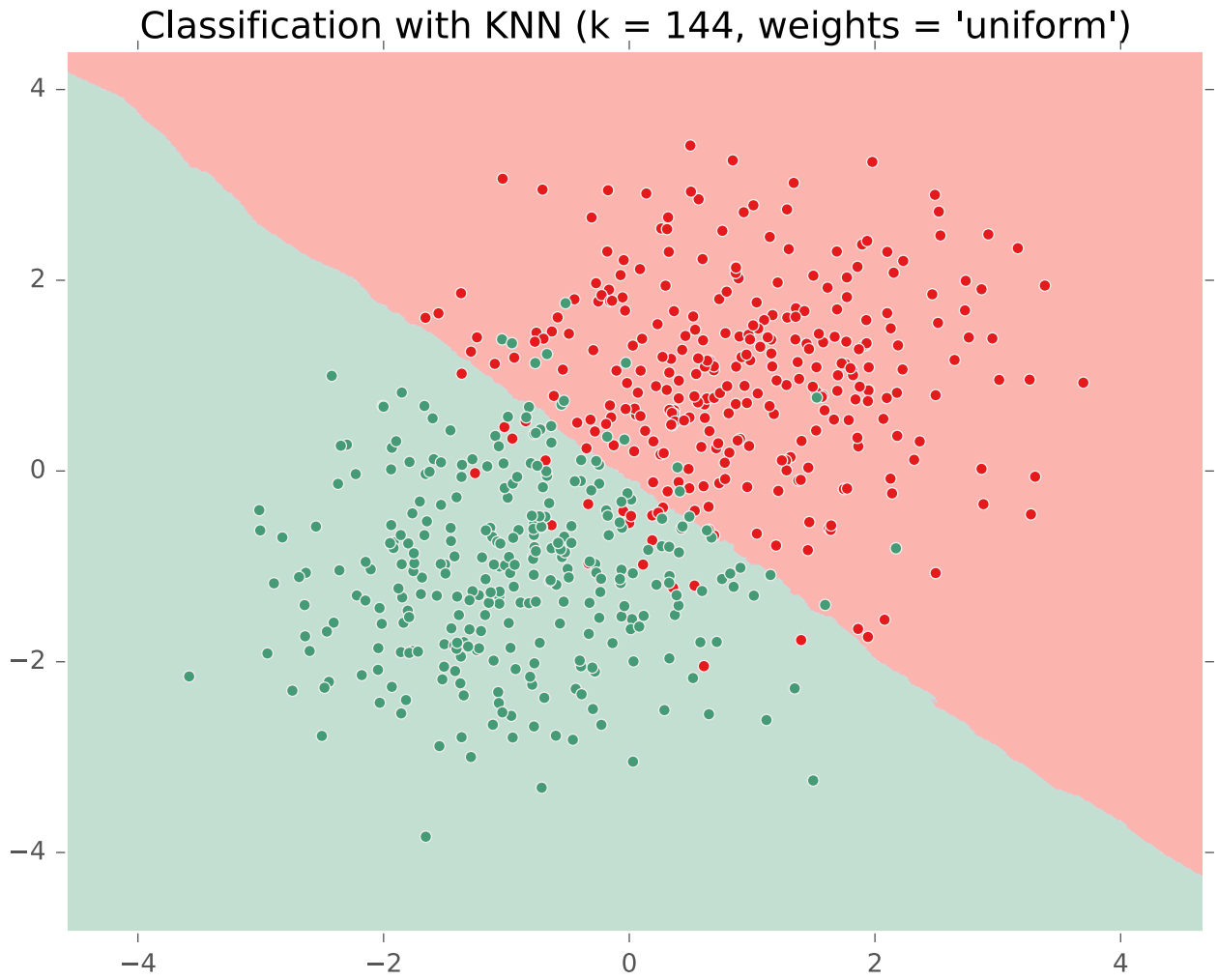
KNN on Gaussian Data



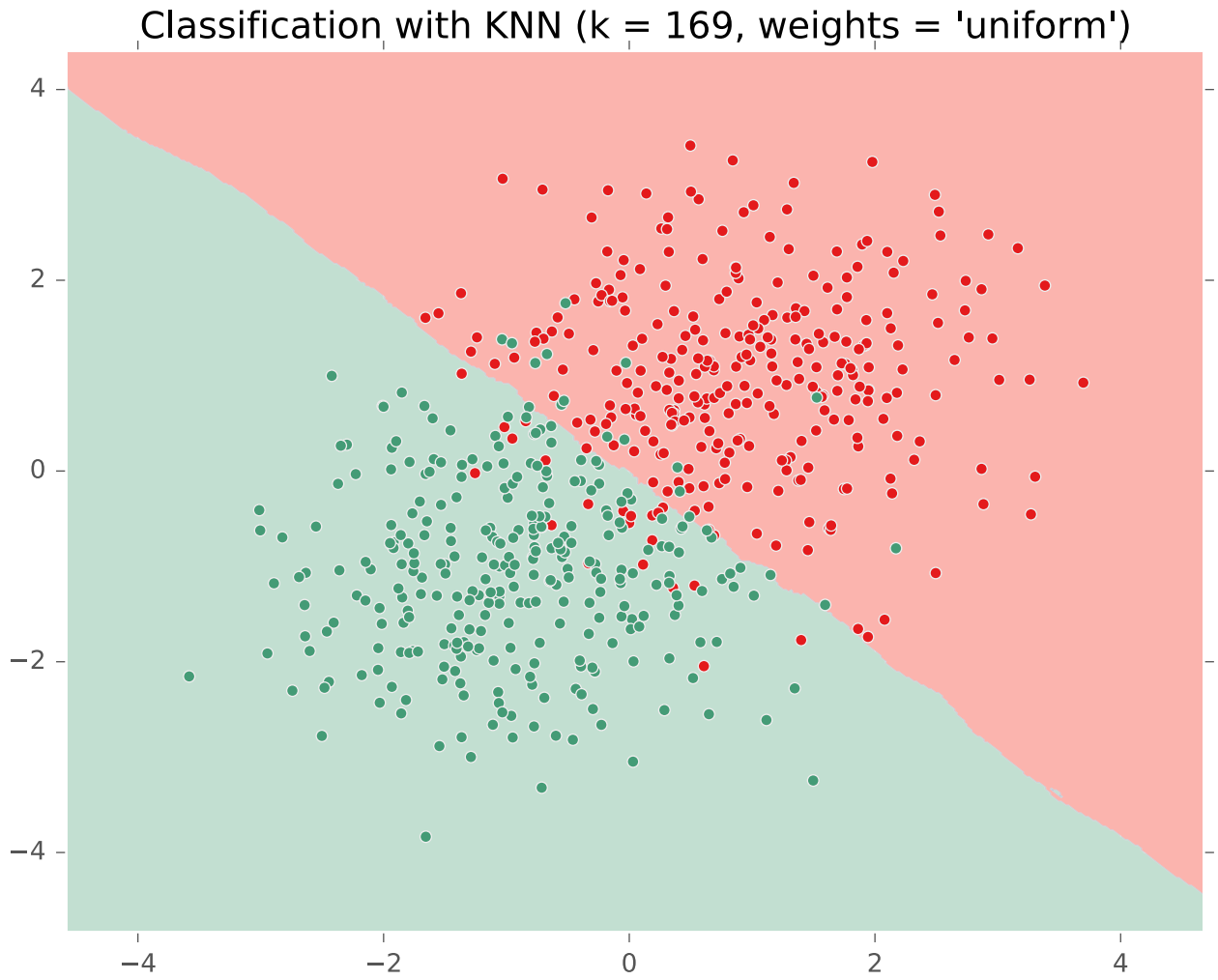
KNN on Gaussian Data



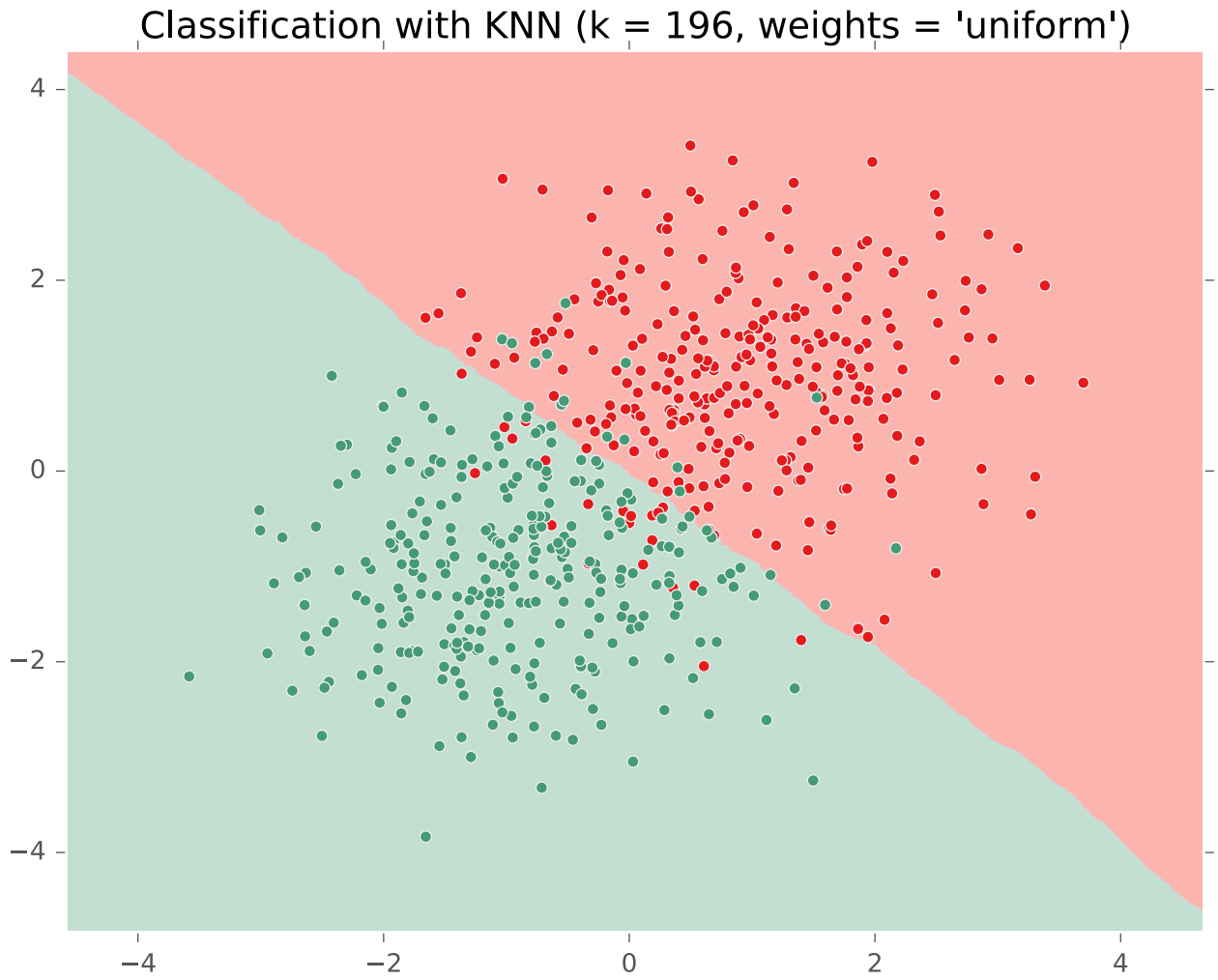
KNN on Gaussian Data



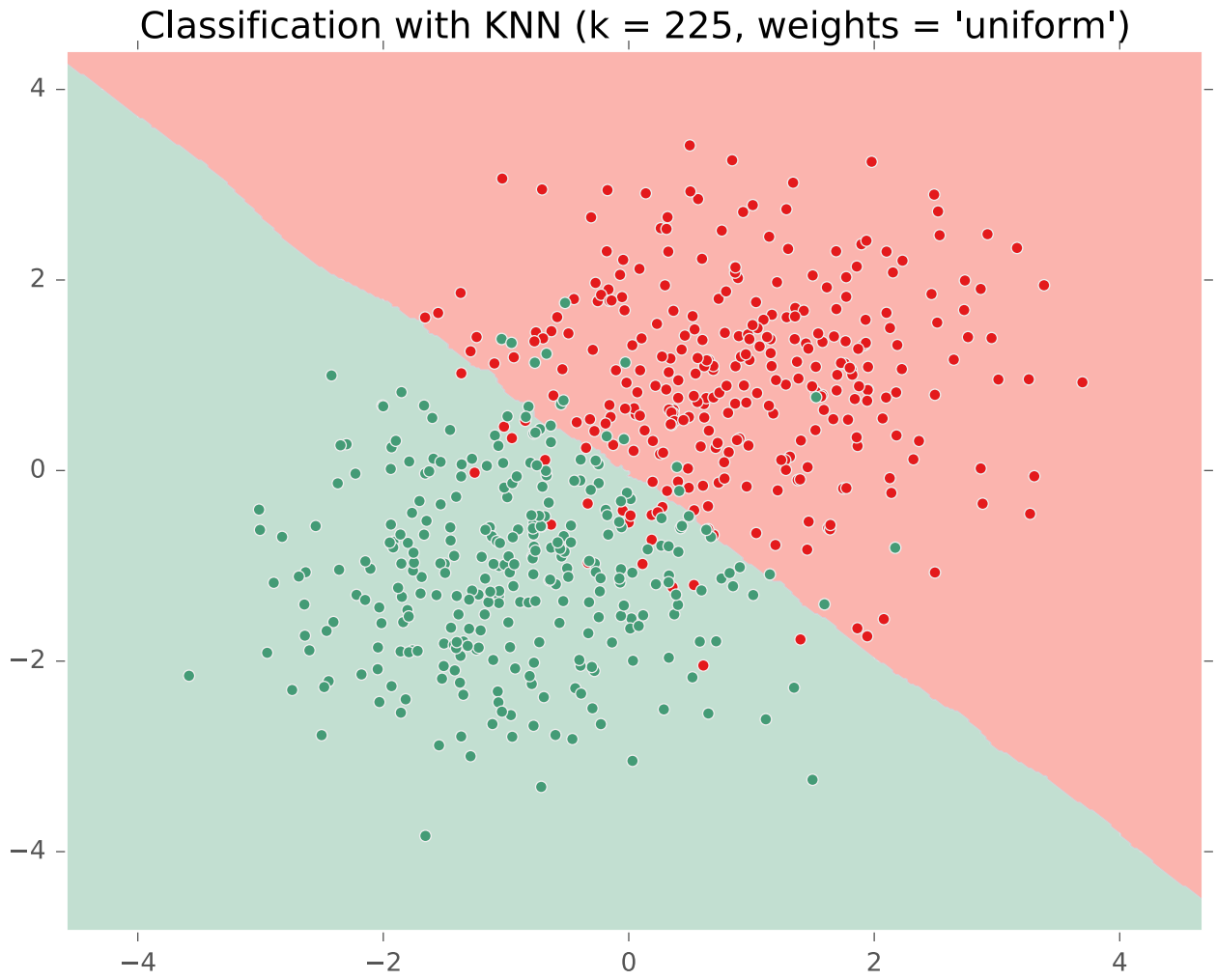
KNN on Gaussian Data



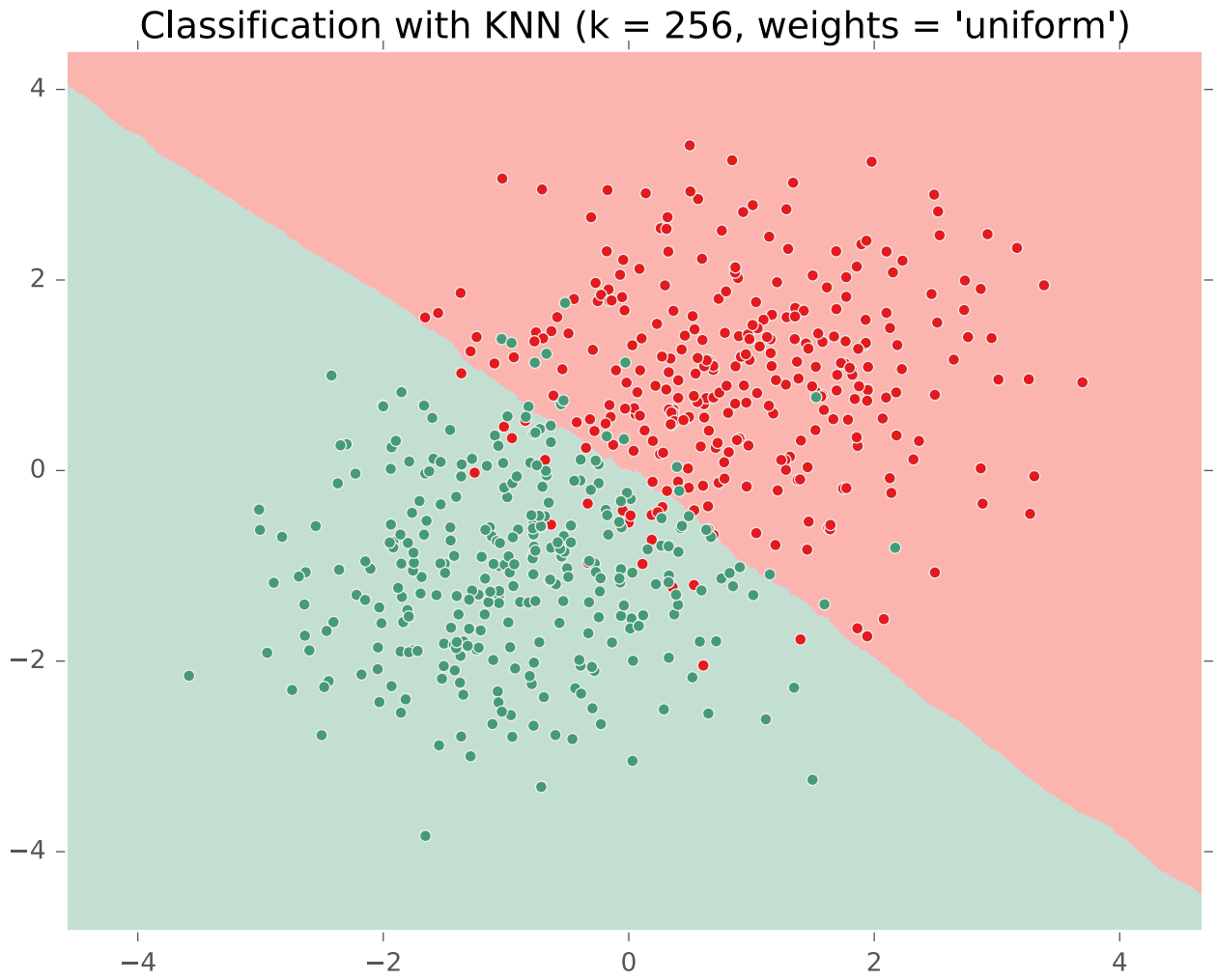
KNN on Gaussian Data



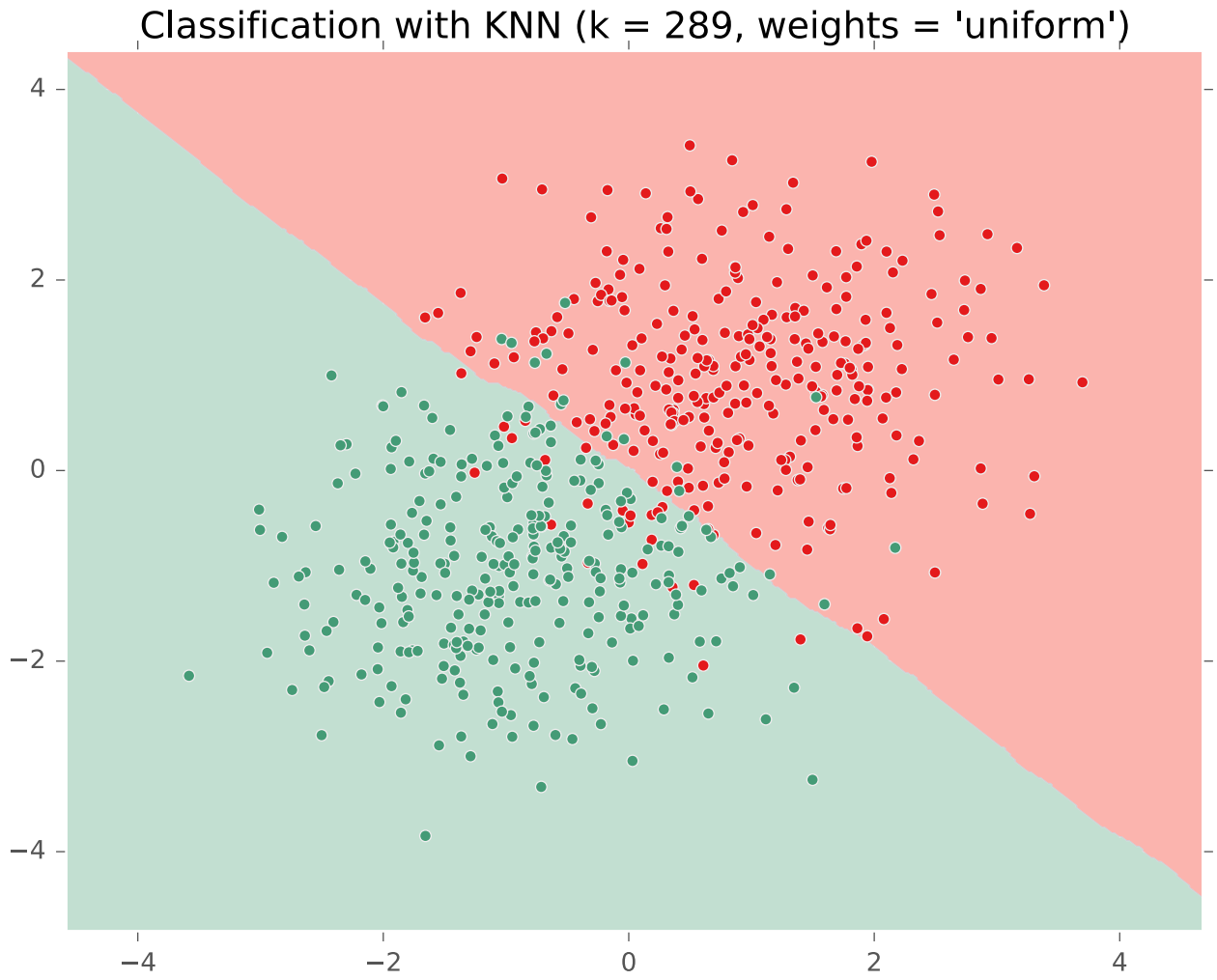
KNN on Gaussian Data



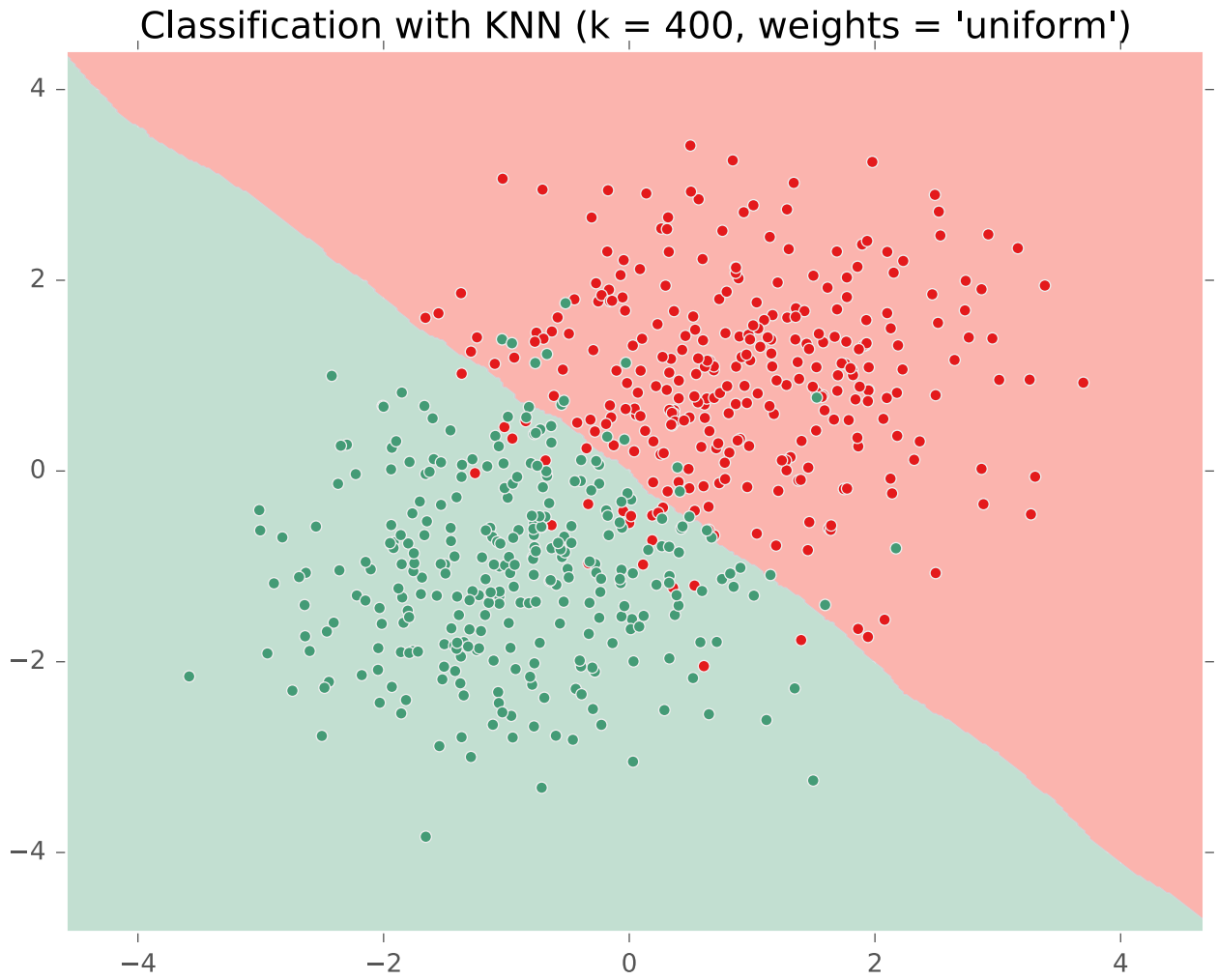
KNN on Gaussian Data



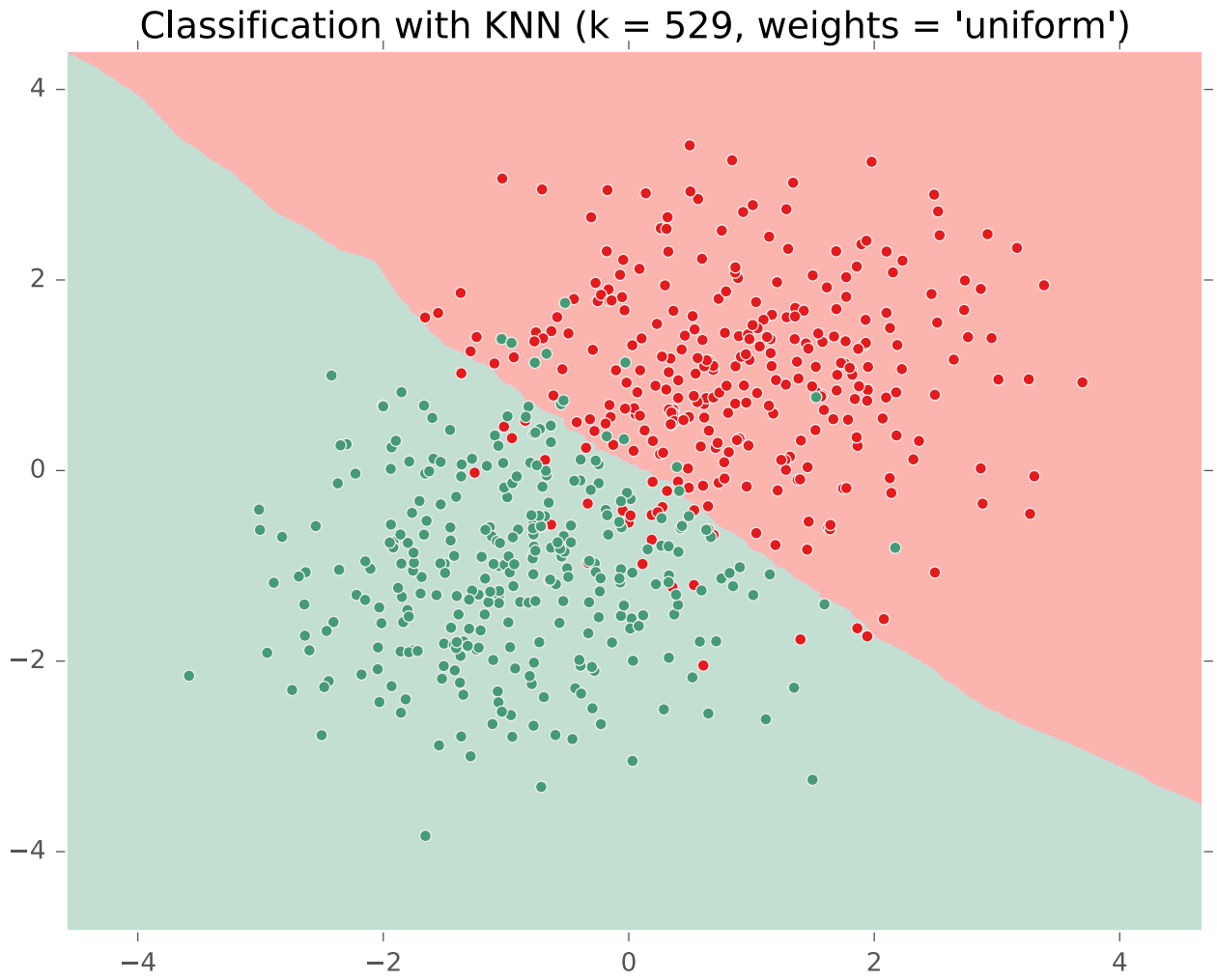
KNN on Gaussian Data



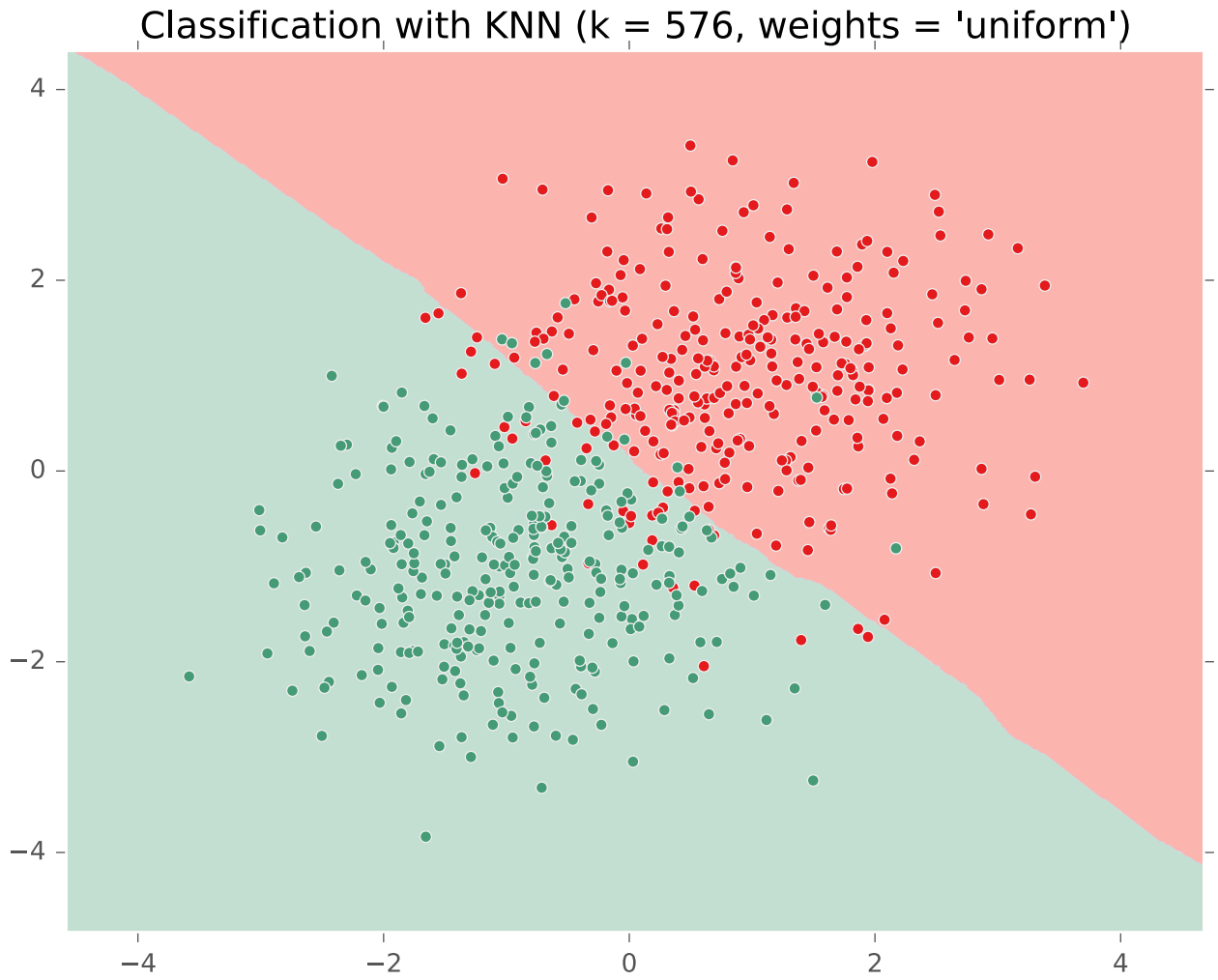
KNN on Gaussian Data



KNN on Gaussian Data



KNN on Gaussian Data



K-NEAREST NEIGHBORS

Questions

- How could k-Nearest Neighbors (KNN) be applied to **regression?**
- Can we do better than majority vote? (e.g. **distance-weighted** KNN)
- Where does the Cover & Hart (1967) **Bayes error rate bound** come from**?**

KNN Learning Objectives

You should be able to…

- Describe a dataset as points in a high dimensional space [CIML]
- Implement k-Nearest Neighbors with O(N) prediction
- Describe the inductive bias of a k-NN classifier and relate it to feature scale [a la. CIML]
- Sketch the decision boundary for a learning algorithm (compare k-NN and DT)
- State Cover & Hart (1967)'s large sample analysis of a nearest neighbor classifier
- Invent "new" k-NN learning algorithms capable of dealing with even k
- Explain computational and geometric examples of the curse of dimensionality

MODEL SELECTION

Model Selection

WARNING:

- In some sense, our discussion of model selection is premature.
- The models we have considered thus far are fairly simple.
- The models and the many decisions available to the data scientist wielding them will grow to be much more complex than what we've seen so far.

Model Selection

Statistics

- *Def*: a **model** defines the data generation process (i.e. a set or family of parametric probability distributions)
- *Def*: **model parameters** are the values that give rise to a particular probability distribution in the model family
- *Def*: **learning** (aka. estimation) is the process of finding the parameters that best fit the data
- *Def*: **hyperparameters** are the parameters of a prior distribution over parameters

Machine Learning

- *Def*: (loosely) a **model** defines the hypothesis space over which learning performs its search
- *Def*: **model parameters** are the numeric values or structure selected by the learning algorithm that give rise to a hypothesis
- *Def*: the **learning algorithm** defines the data-driven search over the hypothesis space (i.e. search for good parameters)
- *Def*: **hyperparameters** are the tunable aspects of the model, that the learning algorithm does *not* select