k-Nearest Neighbors

CENG 499

Introduction to Data Science

Erdoğan Doğdu

The model

- Distance between points
- Points that are close to one another are similar

- Data points and labels
- Label examples:
  - True/False (spam?, poisonous?, enjoyable to watch?)
  - Categories (Ratings: G, PG, PG-13, R, NC-17)
  - Names of presidential candidates
  - Favorite programming language
- Data points: vectors
  - Distance between vectors

The model

- k=3 or 5
- Classify a new data point
  - Find k nearest labeled points and let them vote on the new output

```
def raw_majority_vote(labels):
    votes = Counter(labels)
    winner, _ = votes.most_common(1)[0]
    return winner
```

What if there are more than one?

The model

- Reduce *k* until we find a winner

```
def majority_vote(labels):
    """assumes that labels are ordered from nearest to farthest"""
    vote_counts = Counter(labels)
    winner, winner_count = vote_counts.most_common(1)[0]
    num_winners = len([count
                       for count in vote_counts.values()
                       if count == winner_count])

    if num_winners == 1:
        return winner                     # unique winner, so return it
    else:
        return majority_vote(labels[:-1]) # try again without the farthest
```

Classifier

```
def knn_classify(k, labeled_points, new_point):
    """each labeled point should be a pair (point, label)"""

    # order the labeled points from nearest to farthest
    by_distance = sorted(labeled_points,
                         key=lambda (point, _): distance(point, new_point))

    # find the labels for the k closest
    k_nearest_labels = [label for _, label in by_distance[:k]]

    # and let them vote
    return majority_vote(k_nearest_labels)
```

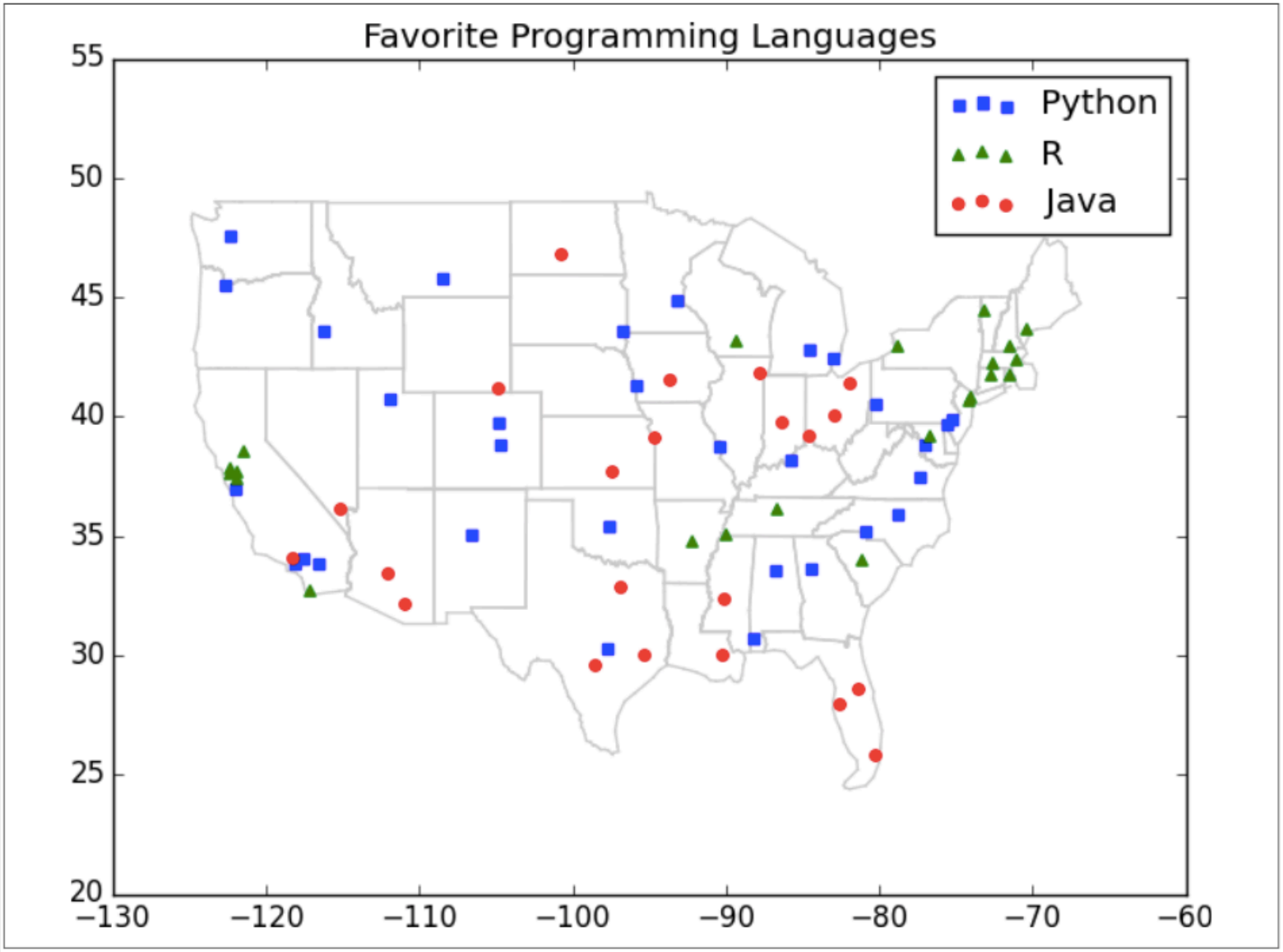
Example: Favorite Languages

- Given

```
# each entry is ([longitude, latitude], favorite_language)

cities = [([-122.3 , 47.53], "Python"),  # Seattle
          ([ -96.85, 32.85], "Java"),    # Austin
          ([ -89.33, 43.13], "R"),       # Madison
          # ... and so on
]
```

- Find the favorite language for new places

Example: Favorite Languages



Try k-NN for all locations

- Predict each city's preferred language using its neighbors other than itself

```
# try several different values for k
for k in [1, 3, 5, 7]:
    num_correct = 0

    for city in cities:
        location, actual_language = city
        other_cities = [other_city
                        for other_city in cities
                        if other_city != city]

        predicted_language = knn_classify(k, other_cities, location)

        if predicted_language == actual_language:
            num_correct += 1

    print k, "neighbor[s]:", num_correct, "correct out of", len(cities)
```

k-NN for all locations

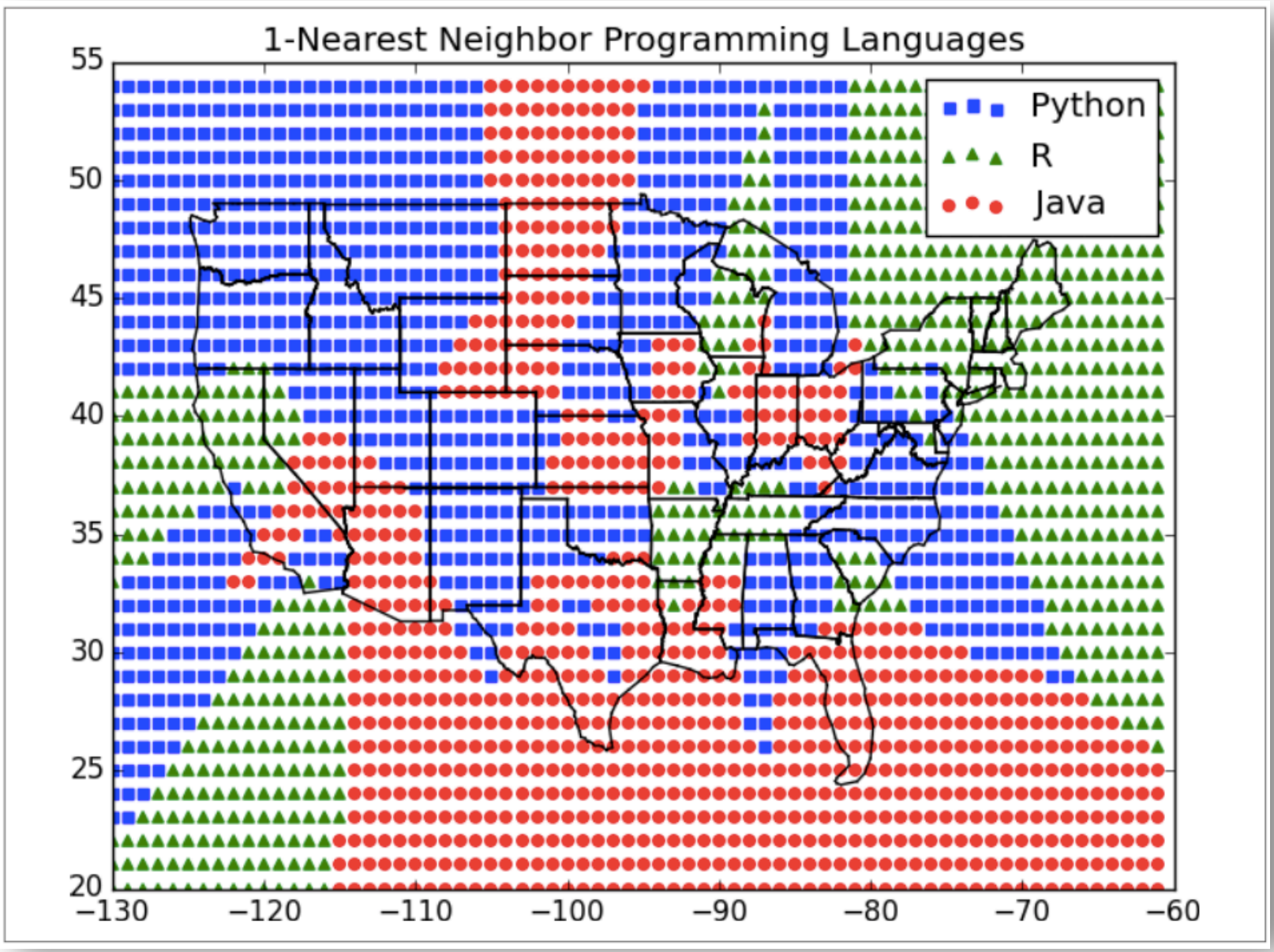
```
1 neighbor[s]: 40 correct out of 75
3 neighbor[s]: 44 correct out of 75
5 neighbor[s]: 41 correct out of 75
7 neighbor[s]: 35 correct out of 75
```

- Best result with k=3 (59% accuracy)

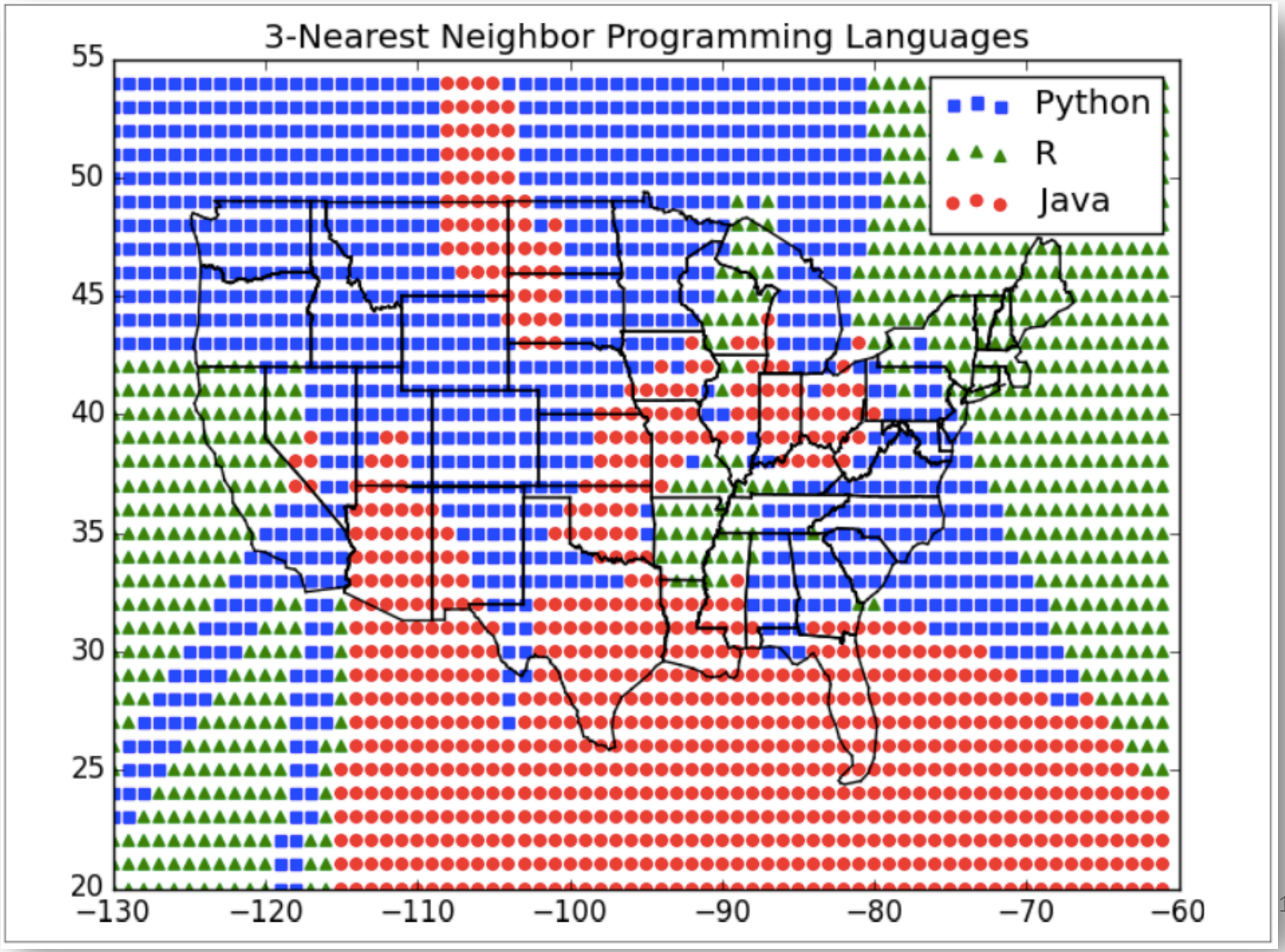
Classify each point in the grid

```
plots = { "Java" : ([], []), "Python" : ([], []), "R" : ([], []) }

k = 1 # or 3, or 5, or ...

for longitude in range(-130, -60):
    for latitude in range(20, 55):
        predicted_language = knn_classify(k, cities, [longitude, latitude])
        plots[predicted_language][0].append(longitude)
        plots[predicted_language][1].append(latitude)
```

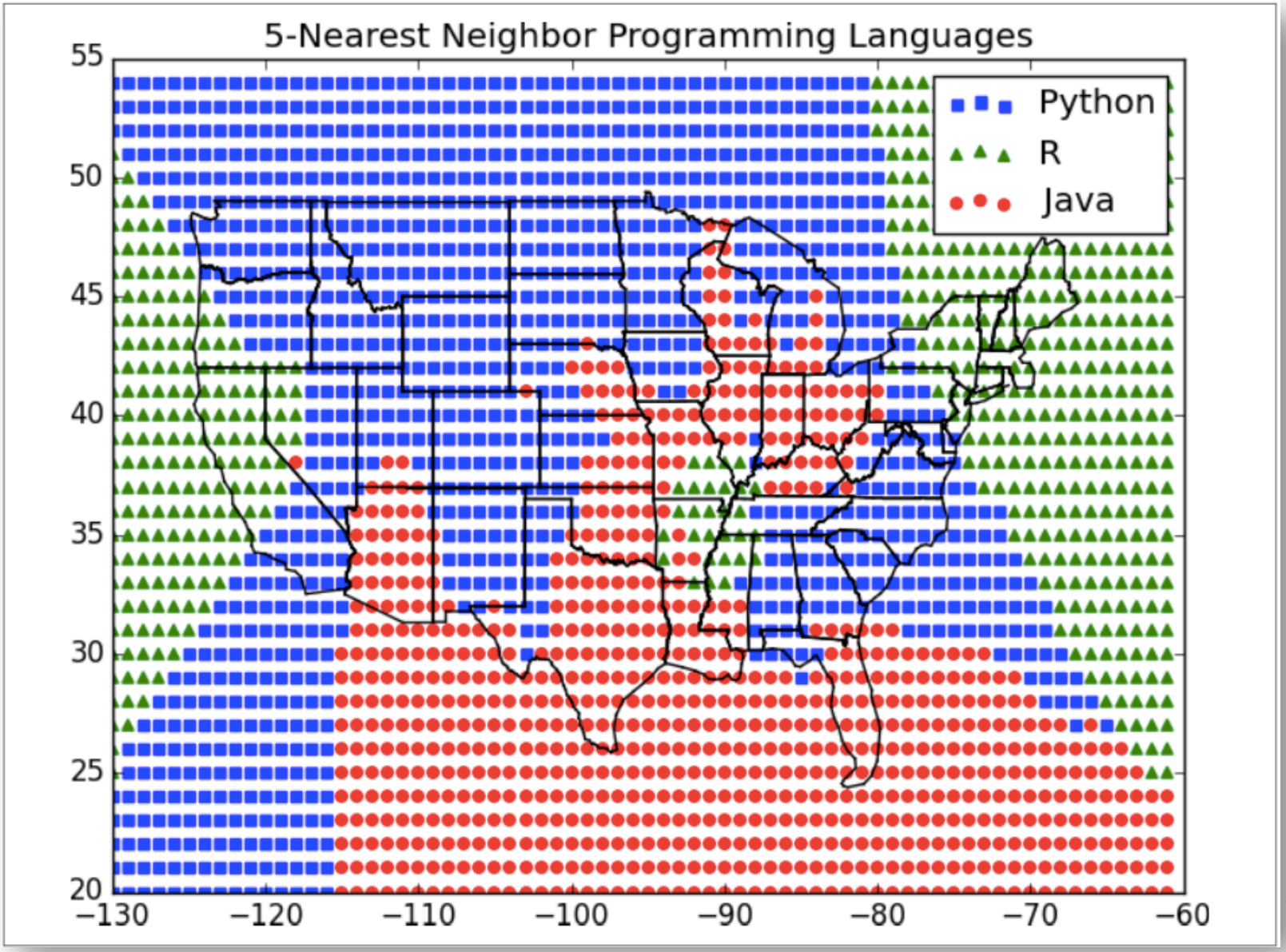
1-NN Results


1-Nearest Neighbor Programming Languages
Python
R
Java
55
50
45
40
35
30
25
20
−130
−120
−110
−100
−90
−80
−70
−60

3-NN Results


3-Nearest Neighbor Programming Languages
Python
R
Java
55
50
45
40
35
30
25
20
−130
−120
−110
−100
−90
−80
−70
−60

5-NN Results


5-Nearest Neighbor Programming Languages
Python
R
Java
55
50
45
40
35
30
25
20
−130
−120
−110
−100
−90
−80
−70
−60