Nearest neighbors algorithm

Toby Dylan Hocking

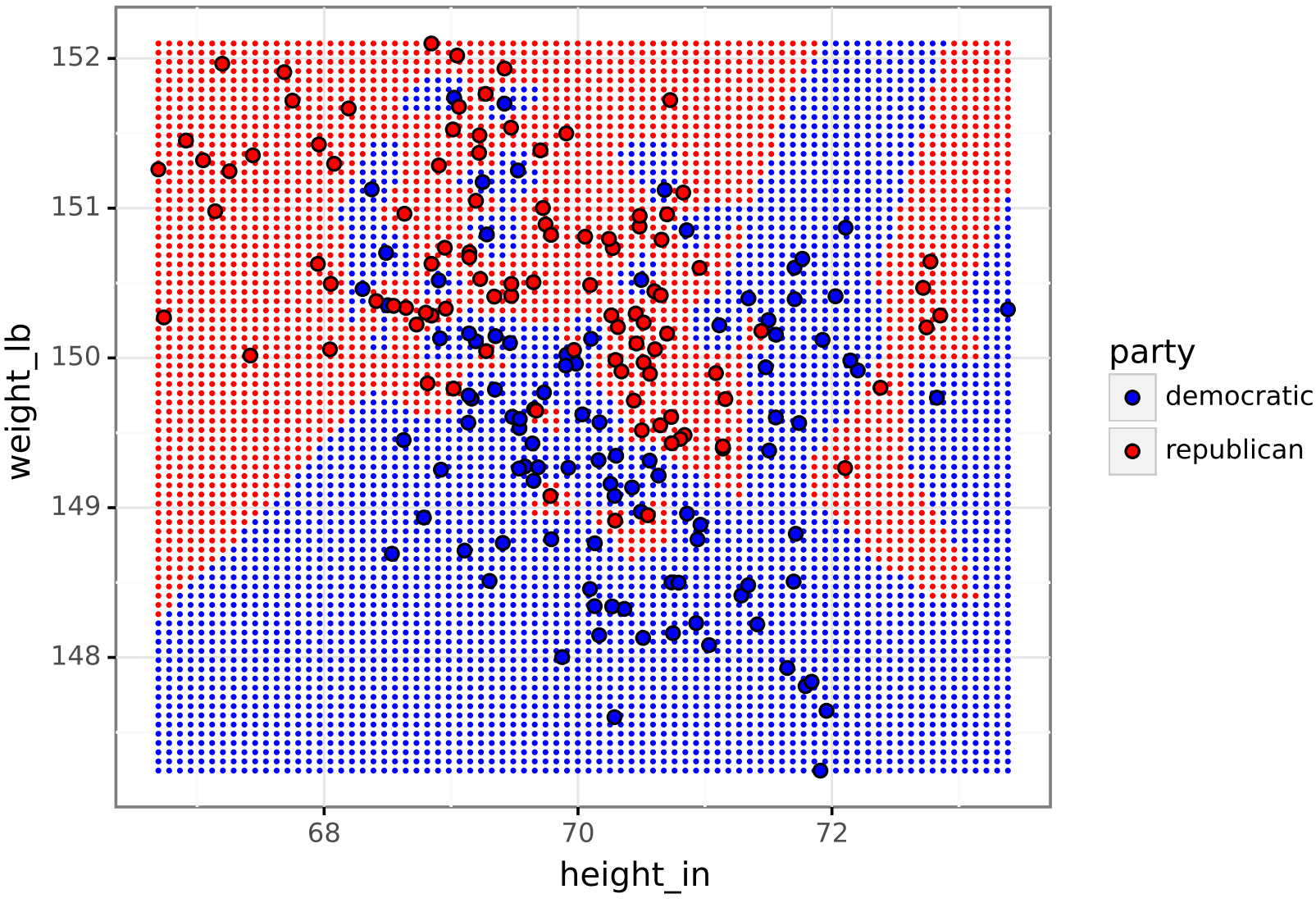
Supervised machine learning

- Goal is to learn a function $f(\mathbf{x}) = y$ where $\mathbf{x}$ is an input/feature vector and y is an output/label.
- $x =$image of digit/clothing, $y \in \{0, \ldots, 9\}$ (ten classes).
- $x =$vector of word counts in email, $y \in \{1, 0\}$ (spam or not).
- Last week we studied two simple machine learning algorithms: nearest neighbors and linear models.
- This week we will study nearest neighbors in depth: distance computations, feature scaling, sensitivity to irrelevant features.

Mixture data table

```
##           party  height_in   weight_lb
## 0    democratic  71.741421  149.565034
## 1    democratic  69.582283  149.275446
## 2    democratic  69.983547  149.961470
## 3    democratic  69.908764  150.021178
## 4    democratic  69.195491  150.111237
## ..          ...        ...         ...
## 195  republican  69.472078  151.537588
## 196  republican  71.140501  149.409036
## 197  republican  70.517269  150.236183
## 198  republican  69.223459  151.486248
## 199  republican  69.019082  149.795387
##
## [200 rows x 3 columns]
```

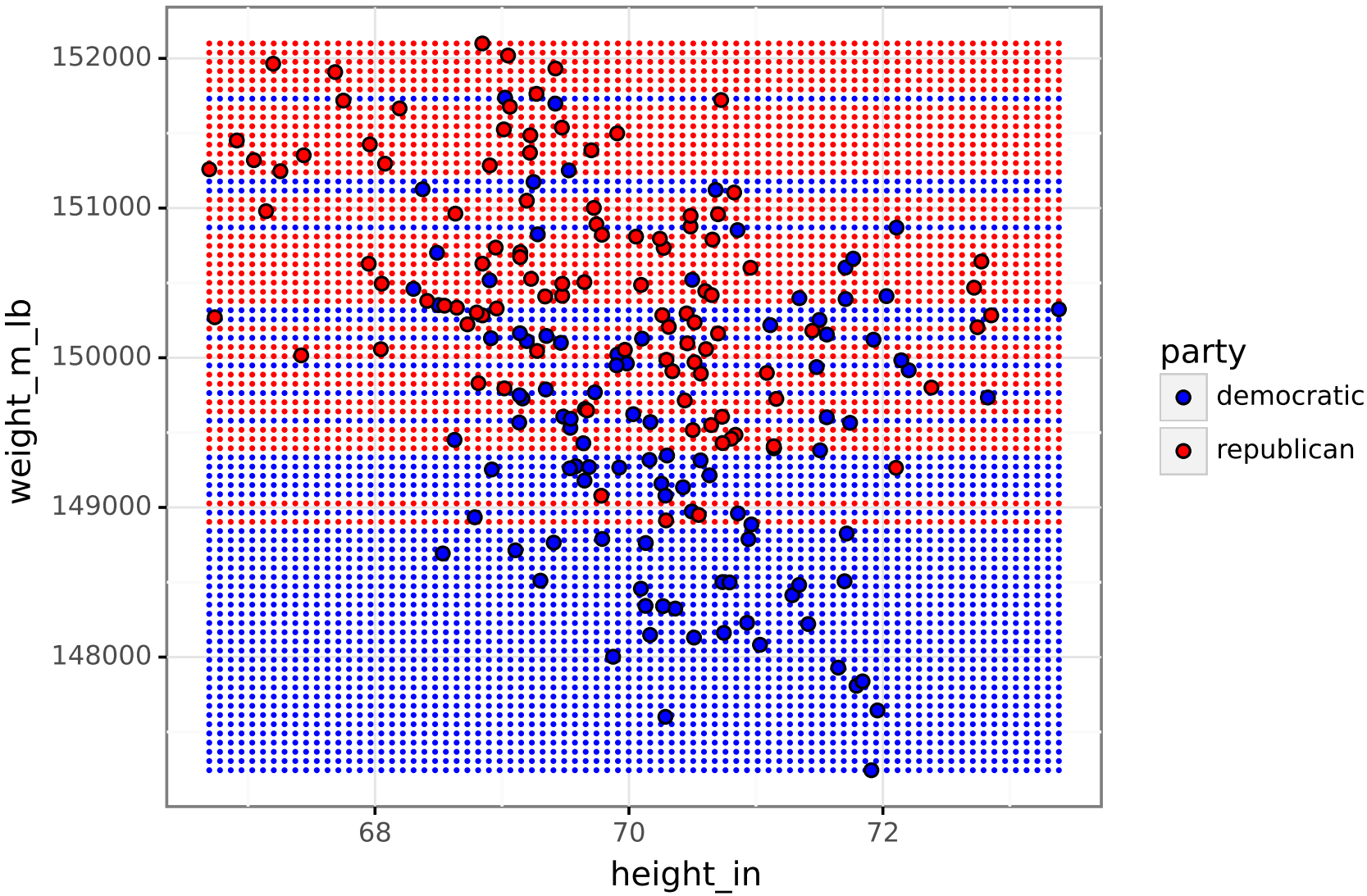
Visualize predictions of 1-nearest neighbor algorithm



Change units of weight to milli-pounds

```
##       height_in    weight_m_lb       party
## 0     71.741421  149565.034079  democratic
## 1     69.582283  149275.445732  democratic
## 2     69.983547  149961.469799  democratic
## 3     69.908764  150021.177667  democratic
## 4     69.195491  150111.237371  democratic
## ..          ...           ...         ...
## 195   69.472078  151537.588208  republican
## 196   71.140501  149409.036272  republican
## 197   70.517269  150236.183248  republican
## 198   69.223459  151486.247543  republican
## 199   69.019082  149795.387192  republican
##
## [200 rows x 3 columns]
```

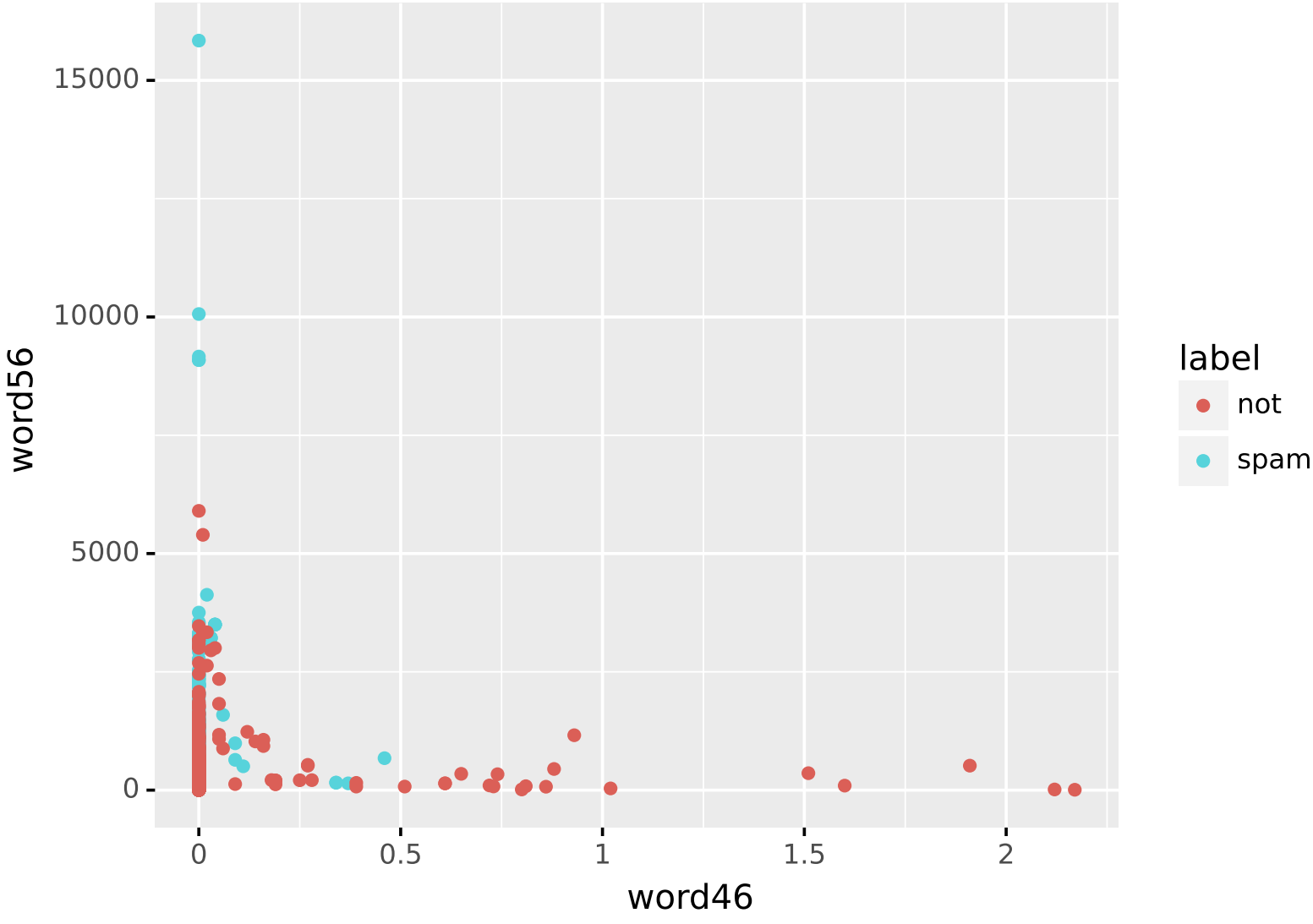
1 nearest neighbor in transformed space ignores height



spam data

```
##          0     1     2   ...   55    56  57
## 0     0.00  0.64  0.64   ...   61   278   1
## 1     0.21  0.28  0.50   ...  101  1028   1
## 2     0.06  0.00  0.71   ...  485  2259   1
## 3     0.00  0.00  0.00   ...   40   191   1
## 4     0.00  0.00  0.00   ...   40   191   1
## ...    ...   ...   ...   ...  ...   ...  ..
## 4596  0.31  0.00  0.62   ...    3    88   0
## 4597  0.00  0.00  0.00   ...    4    14   0
## 4598  0.30  0.00  0.30   ...    6   118   0
## 4599  0.96  0.00  0.00   ...    5    78   0
## 4600  0.00  0.00  0.65   ...    5    40   0
##
## [4601 rows x 58 columns]
```

Two columns from spam data have different scales



Correcting for feature scale

- Nearest neighbor algorithm is sensitive to feature scales.
- Features with larger values are artificially more important.
- Before learning need to scale each feature (subtract mean, divide by standard deviation).

What if some features are not important?

- For some problems there may be features which are not relevant to predicting the label.
- Example from biology: predict whether or not a person has sickle cell disease, using various physical attributes: genetics, height, weight, eye color, etc.
- Sickle cell disease happens when there are mutations in the beta-globin gene, so other features are totally irrelevant.
- If you know which features are irrelevant, then exclude them from your feature vector $\mathbf{x}$.
- If you do not know, then the irrelevant features will reduce your accuracy.

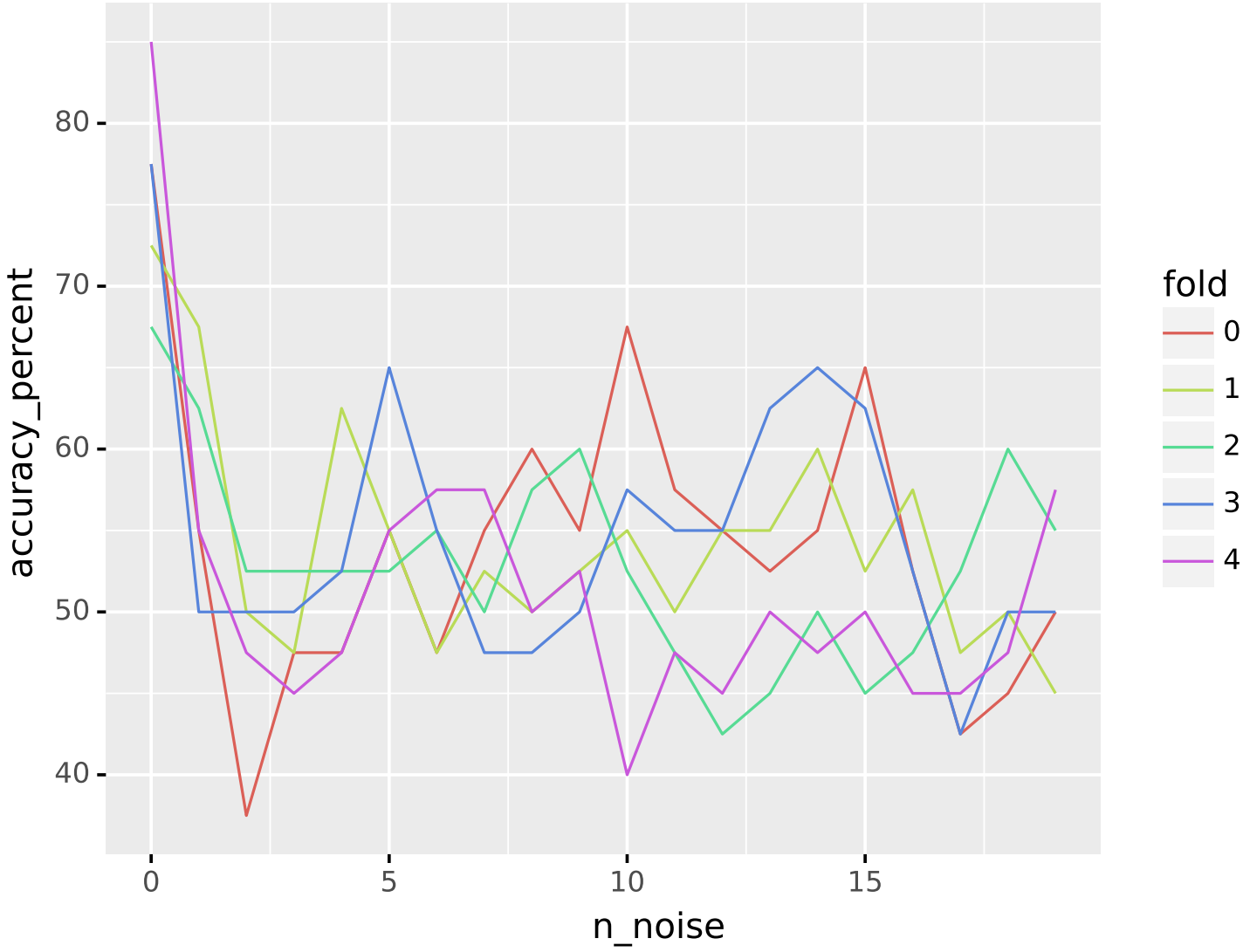
Add one noise feature using np.random.randn

```
##            party  height_in   weight_lb      noise1
## 0     democratic  71.741421  149.565034 -142.609849
## 1     democratic  69.582283  149.275446  137.697161
## 2     democratic  69.983547  149.961470   29.210544
## 3     democratic  69.908764  150.021178   49.894752
## 4     democratic  69.195491  150.111237  109.949784
## ..           ...        ...         ...         ...
## 195   republican  69.472078  151.537588 -113.619320
## 196   republican  71.140501  149.409036   94.891968
## 197   republican  70.517269  150.236183  -73.271338
## 198   republican  69.223459  151.486248  -33.850543
## 199   republican  69.019082  149.795387  136.602786
##
## [200 rows x 4 columns]
```

Add more noise features

```
##           party  height_in  ...       noise1      noise2
## 0    democratic  71.741421  ... -142.609849  102.363932
## 1    democratic  69.582283  ...  137.697161  -86.370867
## 2    democratic  69.983547  ...   29.210544 -166.412764
## 3    democratic  69.908764  ...   49.894752   87.215300
## 4    democratic  69.195491  ...  109.949784   78.125166
## ..          ...        ...  ...         ...         ...
## 195  republican  69.472078  ... -113.619320 -122.873409
## 196  republican  71.140501  ...   94.891968   37.139980
## 197  republican  70.517269  ...  -73.271338  -39.307499
## 198  republican  69.223459  ...  -33.850543   77.244742
## 199  republican  69.019082  ...  136.602786  -85.937556
##
## [200 rows x 5 columns]
```

Simulation of test accuracy as noise features are added



Complexity analysis / pseudo code

- Let there be n rows and p columns in the train set input/feature matrix, with k neighbors.
- To compute the distance between a pair of data points/rows, it takes $O(p)$ time (for loop over columns/features).
- To compute the n distances between all of the train data and a new test data point, it takes $O(np)$ time (for loop over train set).
- Then you have to sort the n distances to find the smallest k distances, $O(n \log n)$.
- Finally you compute the predicted probability in a for loop over the nearest k neighbors.
- Overall time complexity $O(np + n \log n)$.