

Machine learning with scikit-learn

SUPERVISED LEARNING WITH SCIKIT-LEARN



George Boorman

Core Curriculum Manager, DataCamp

What is machine learning?

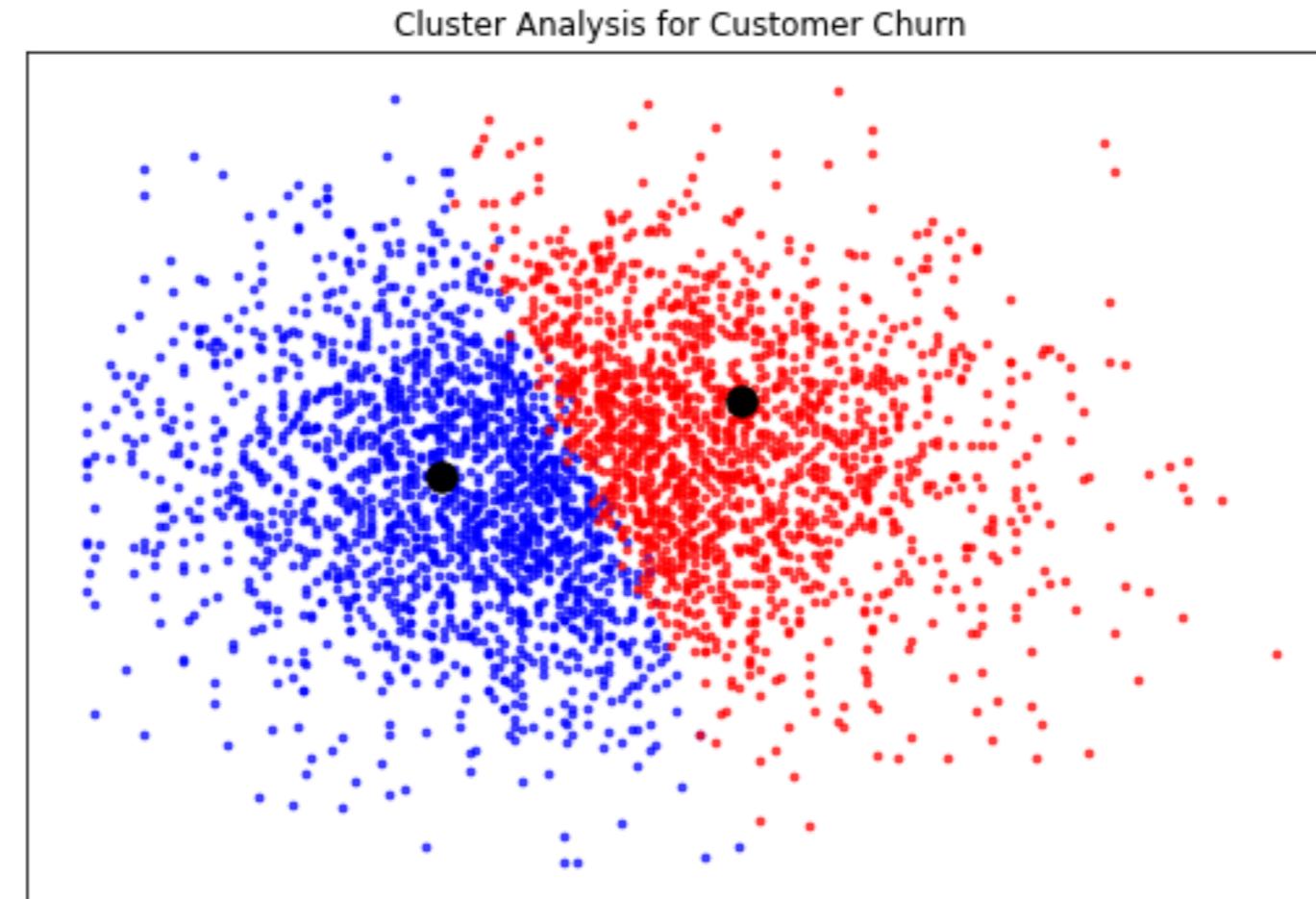
- Machine learning is the process whereby:
 - Computers are given the ability to learn to make decisions from data
 - without being explicitly programmed!

Examples of machine learning



Unsupervised learning

- Uncovering hidden patterns from unlabeled data
- Example:
 - Grouping customers into distinct categories (Clustering)



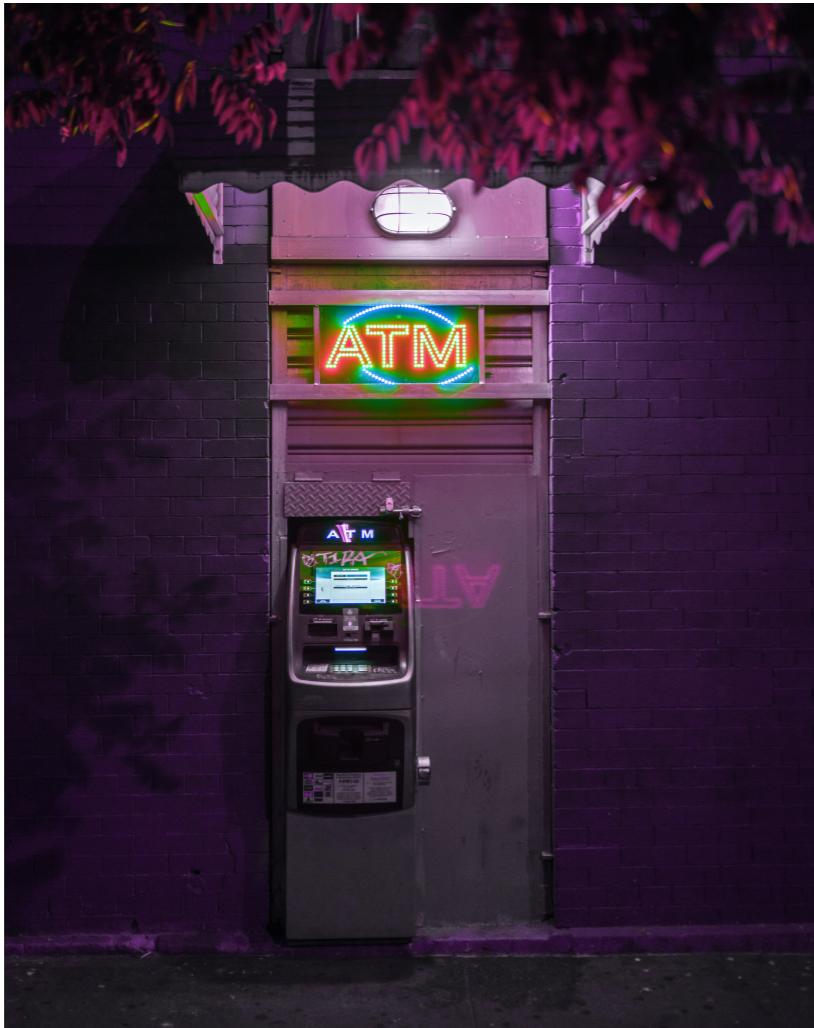
Supervised learning

- The predicted values are known
- Aim: Predict the target values of unseen data, given the features

	Features					Target variable
	points_per_game	assists_per_game	rebounds_per_game	steals_per_game	blocks_per_game	position
0	26.9	6.6	4.5	1.1	0.4	Point Guard
1	13	1.7	4	0.4	1.3	Center
2	17.6	2.3	7.9	1.00	0.8	Power Forward
3	22.6	4.5	4.4	1.2	0.4	Shooting Guard

Types of supervised learning

- Classification: Target variable consists of categories
- Regression: Target variable is continuous



Naming conventions

- Feature = predictor variable = independent variable
- Target variable = dependent variable = response variable

Before you use supervised learning

- Requirements:
 - No missing values
 - Data in numeric format
 - Data stored in pandas DataFrame or NumPy array
- Perform Exploratory Data Analysis (EDA) first

scikit-learn syntax

```
from sklearn.module import Model  
  
model = Model()  
  
model.fit(X, y)  
  
predictions = model.predict(X_new)  
print(predictions)
```

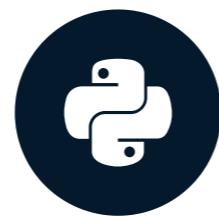
```
array([0, 0, 0, 0, 1, 0])
```

Let's practice!

SUPERVISED LEARNING WITH SCIKIT-LEARN

The classification challenge

SUPERVISED LEARNING WITH SCIKIT-LEARN



George Boorman

Core Curriculum Manager, DataCamp

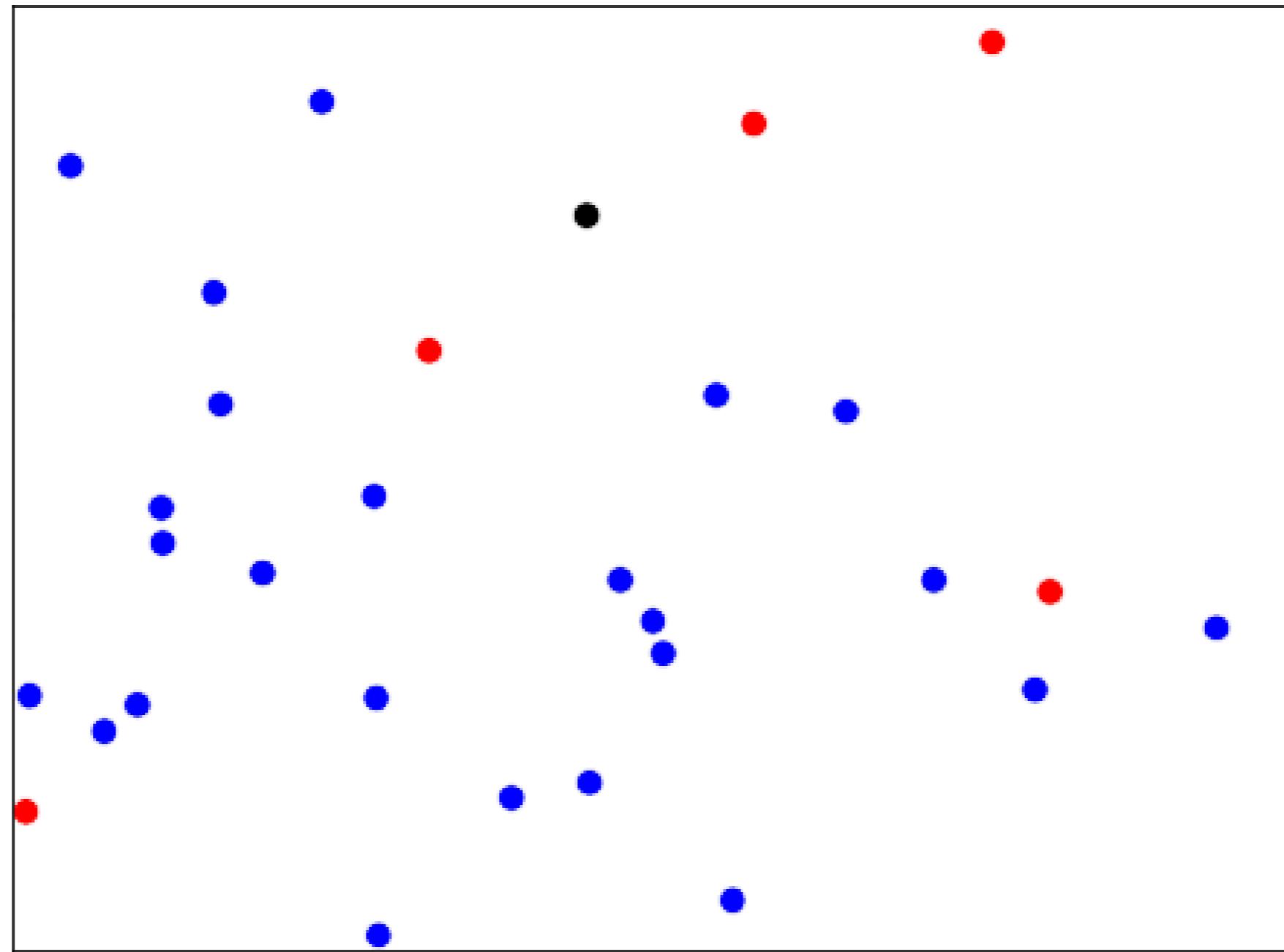
Classifying labels of unseen data

1. Build a model
 2. Model learns from the labeled data we pass to it
 3. Pass unlabeled data to the model as input
 4. Model predicts the labels of the unseen data
- Labeled data = training data

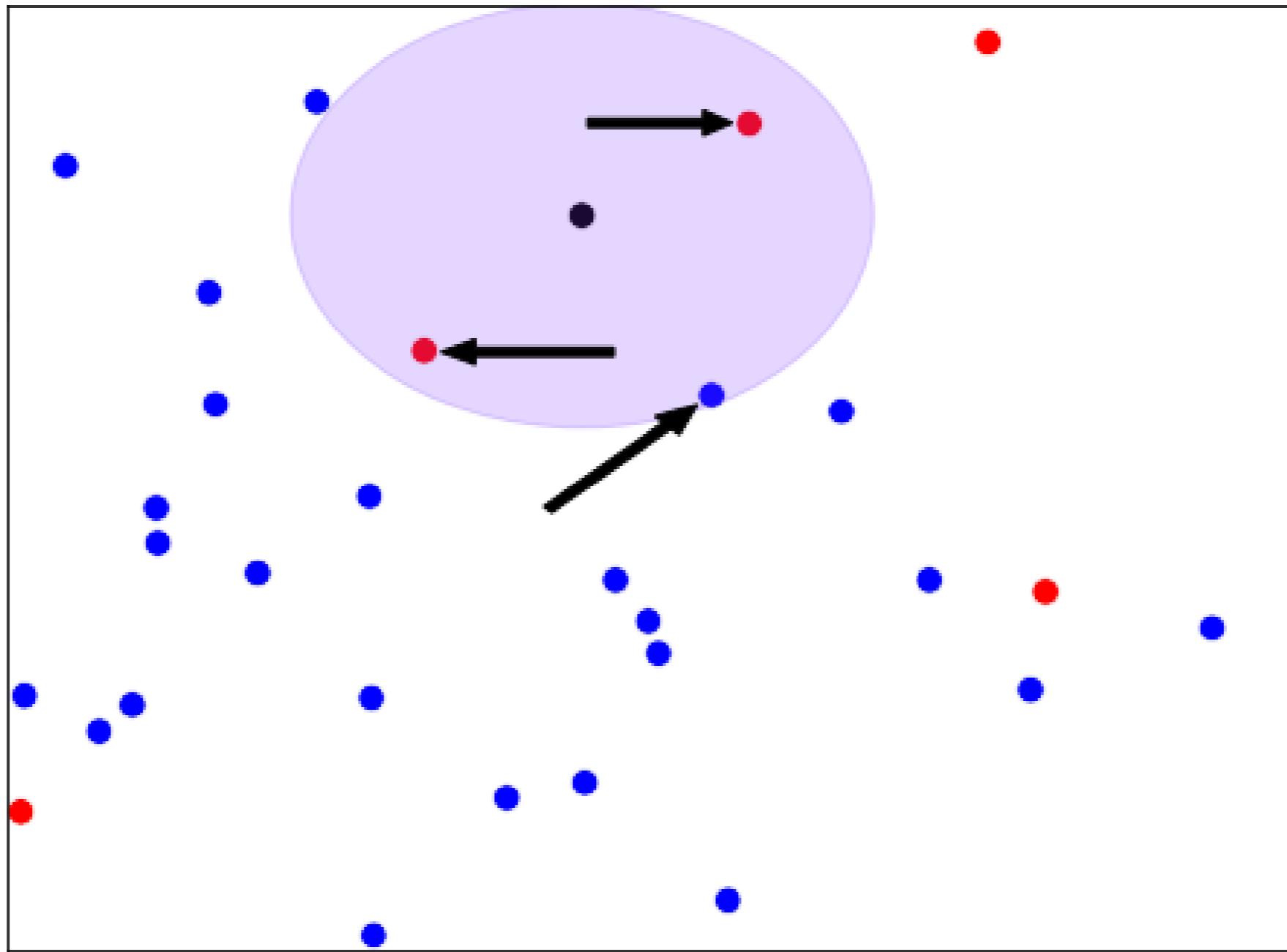
k-Nearest Neighbors

- Predict the label of a data point by
 - Looking at the k closest labeled data points
 - Taking a majority vote

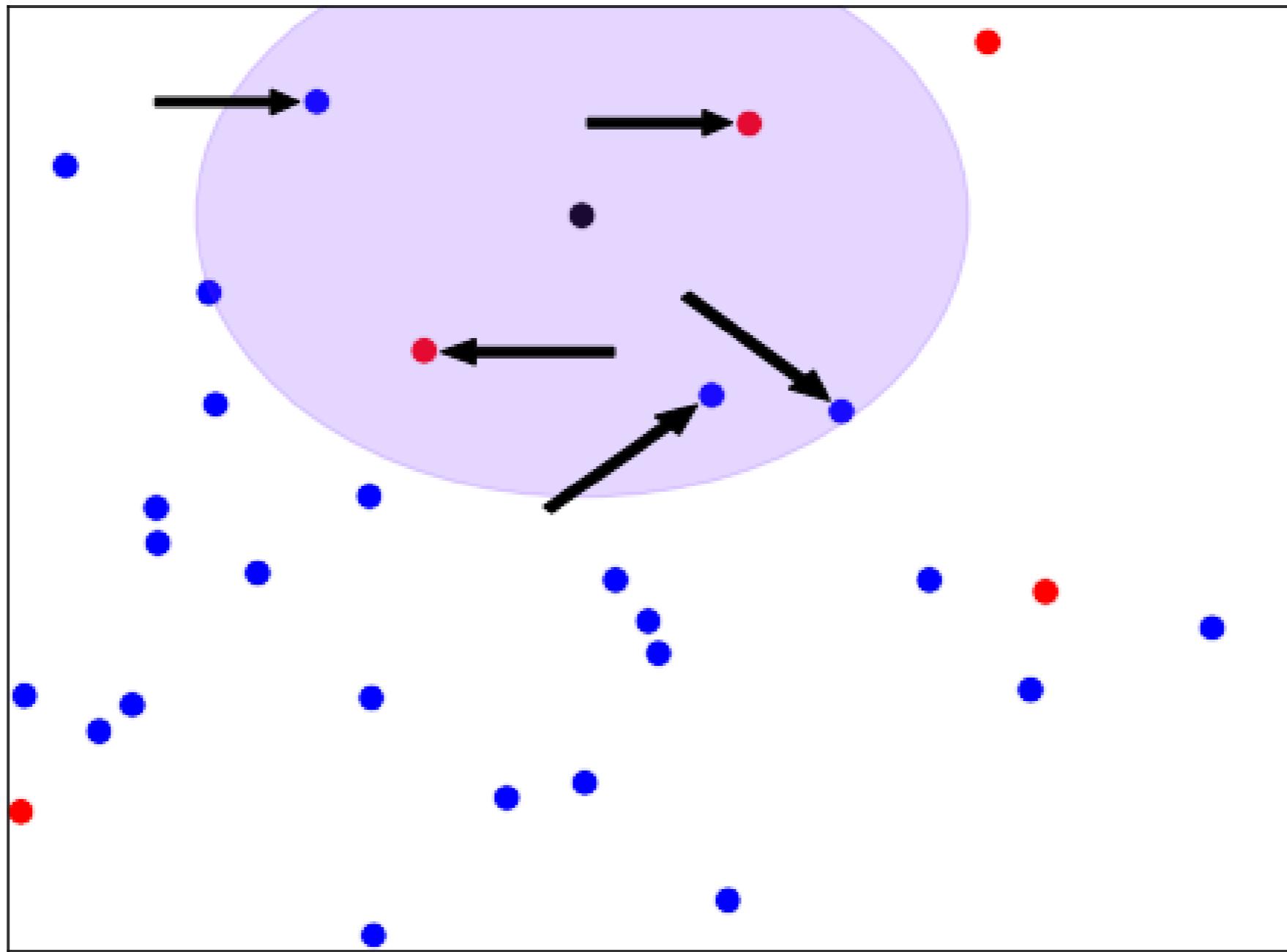
k-Nearest Neighbors



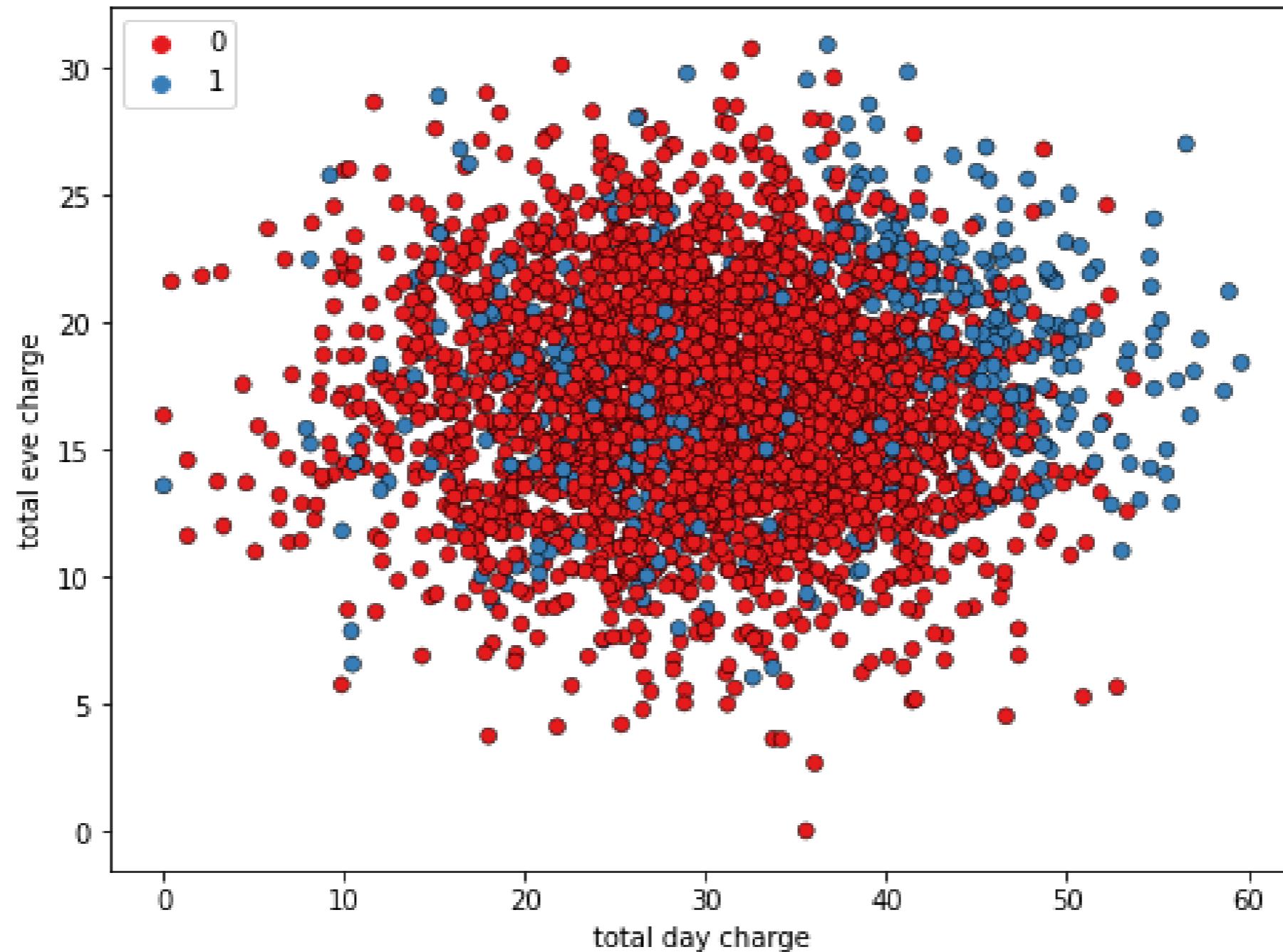
k-Nearest Neighbors



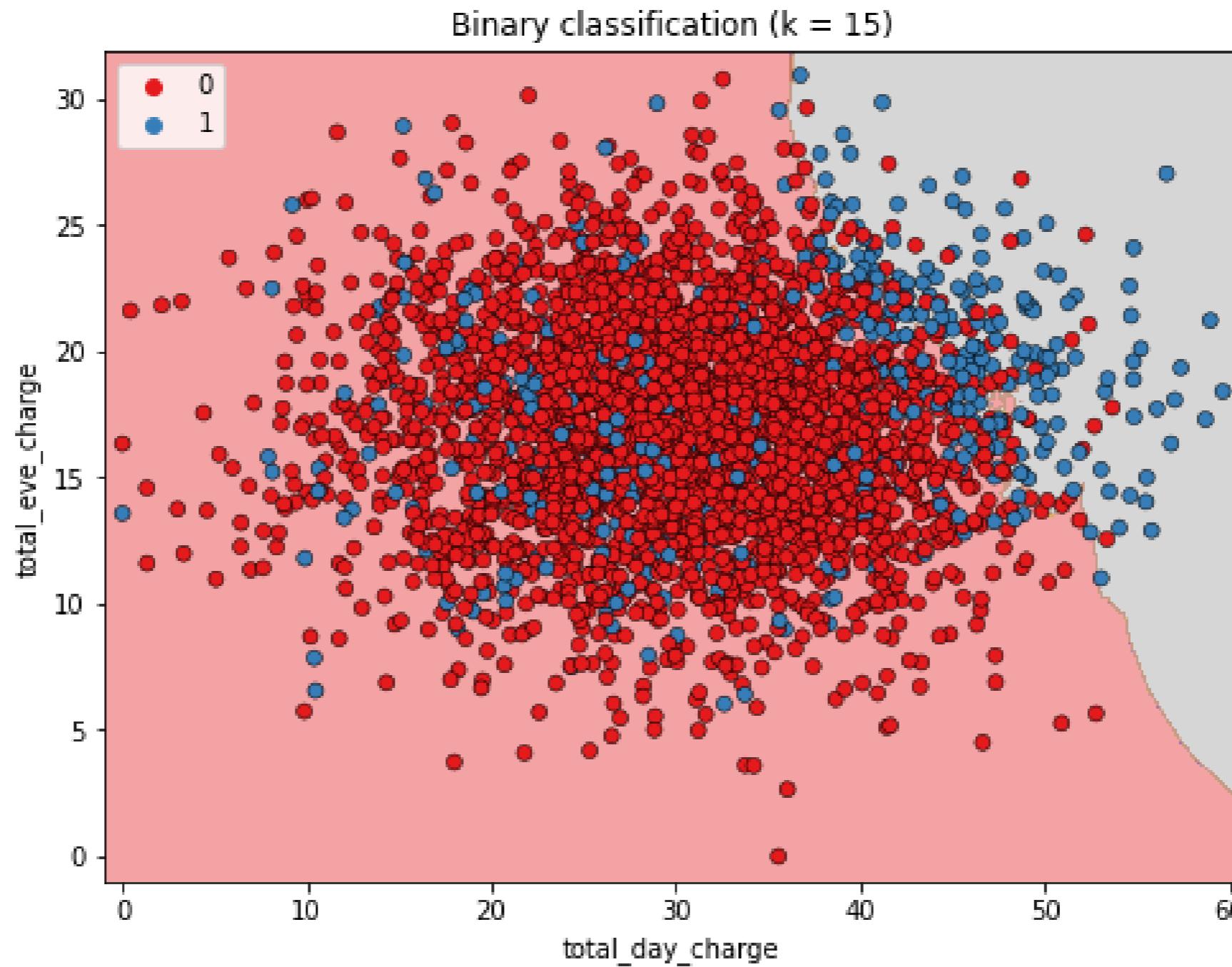
k-Nearest Neighbors



KNN Intuition



KNN Intuition



Using scikit-learn to fit a classifier

```
from sklearn.neighbors import KNeighborsClassifier  
X = churn_df[["total_day_charge", "total_eve_charge"]].values  
y = churn_df["churn"].values  
print(X.shape, y.shape)
```

```
(333, 2), (333,)
```

```
knn = KNeighborsClassifier(n_neighbors=15)  
knn.fit(X, y)
```

Predicting on unlabeled data

```
X_new = np.array([[56.8, 17.5],  
                  [24.4, 24.1],  
                  [50.1, 10.9]])  
  
print(X_new.shape)
```

```
(3, 2)
```

```
predictions = knn.predict(X_new)  
print('Predictions: {}'.format(predictions))
```

```
Predictions: [1 0 0]
```

Let's practice!

SUPERVISED LEARNING WITH SCIKIT-LEARN

Measuring model performance

SUPERVISED LEARNING WITH SCIKIT-LEARN



George Boorman

Core Curriculum Manager, DataCamp

Measuring model performance

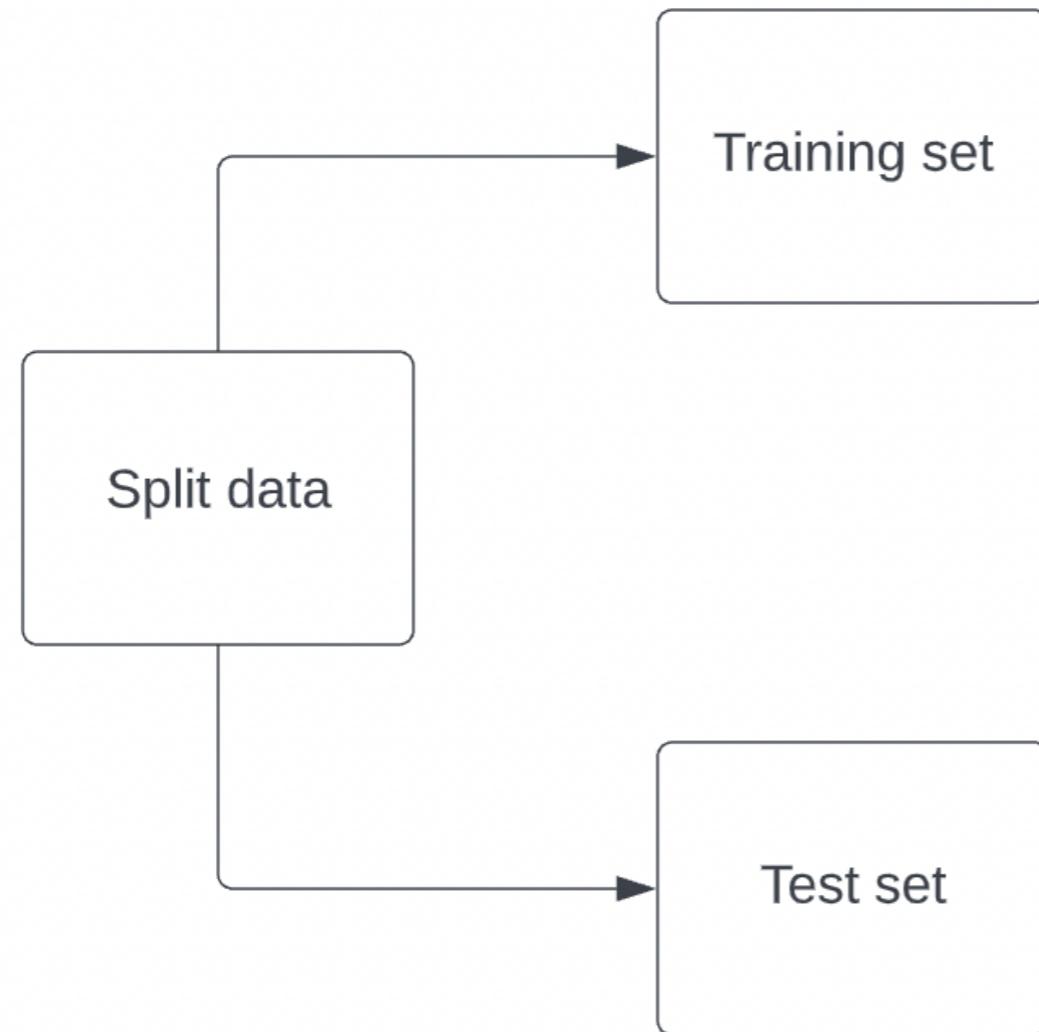
- In classification, accuracy is a commonly used metric
- Accuracy:

$$\frac{\text{correct predictions}}{\text{total observations}}$$

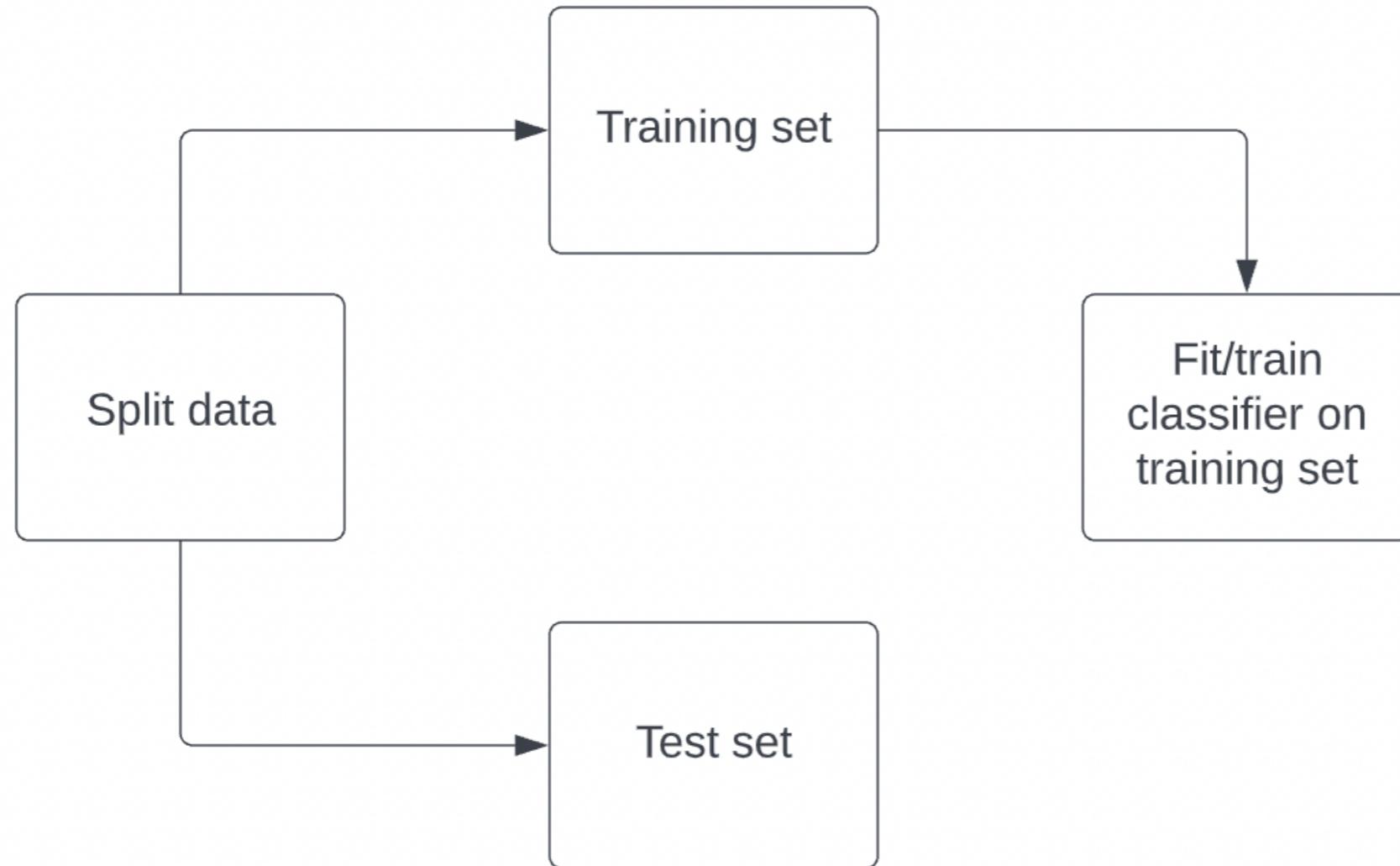
Measuring model performance

- How do we measure accuracy?
- Could compute accuracy on the data used to fit the classifier
- NOT indicative of ability to generalize

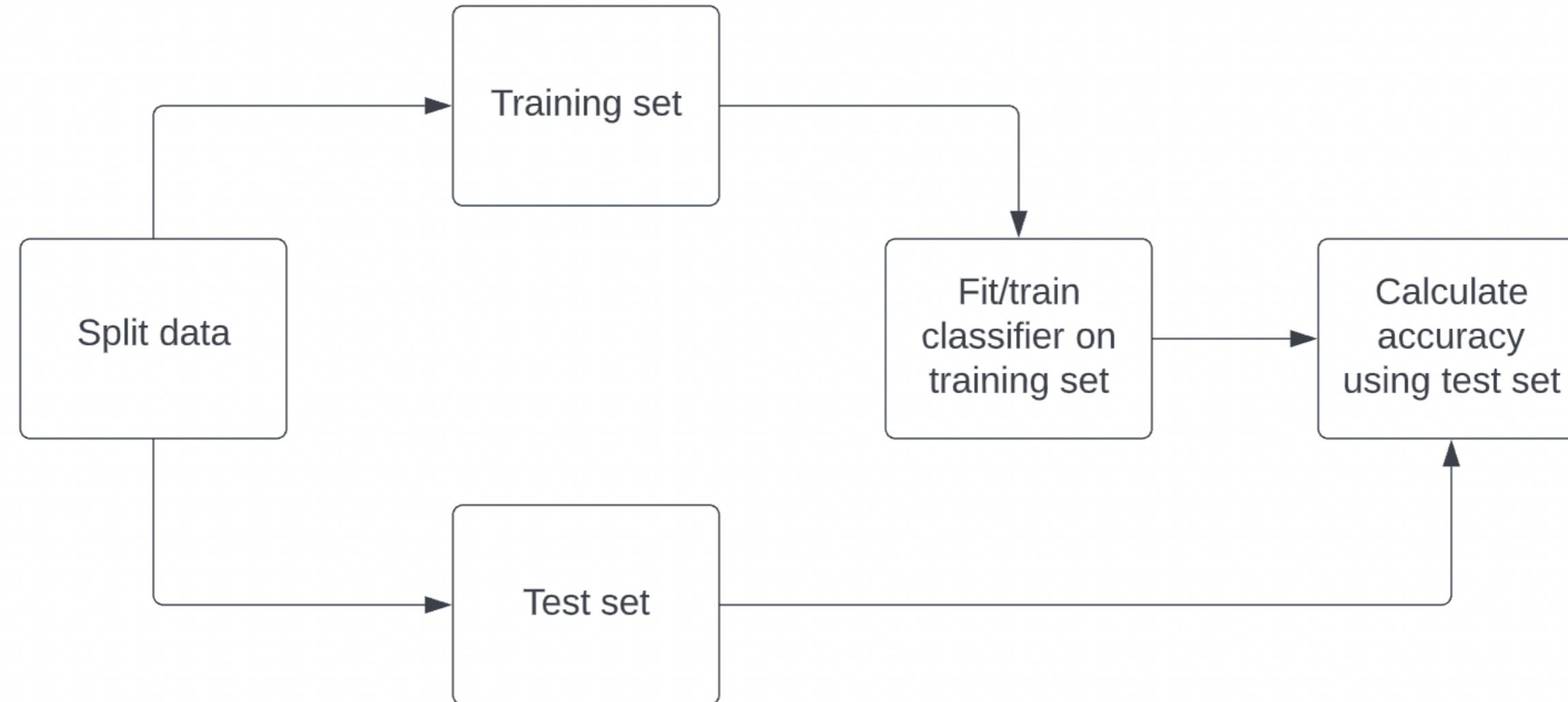
Computing accuracy



Computing accuracy



Computing accuracy



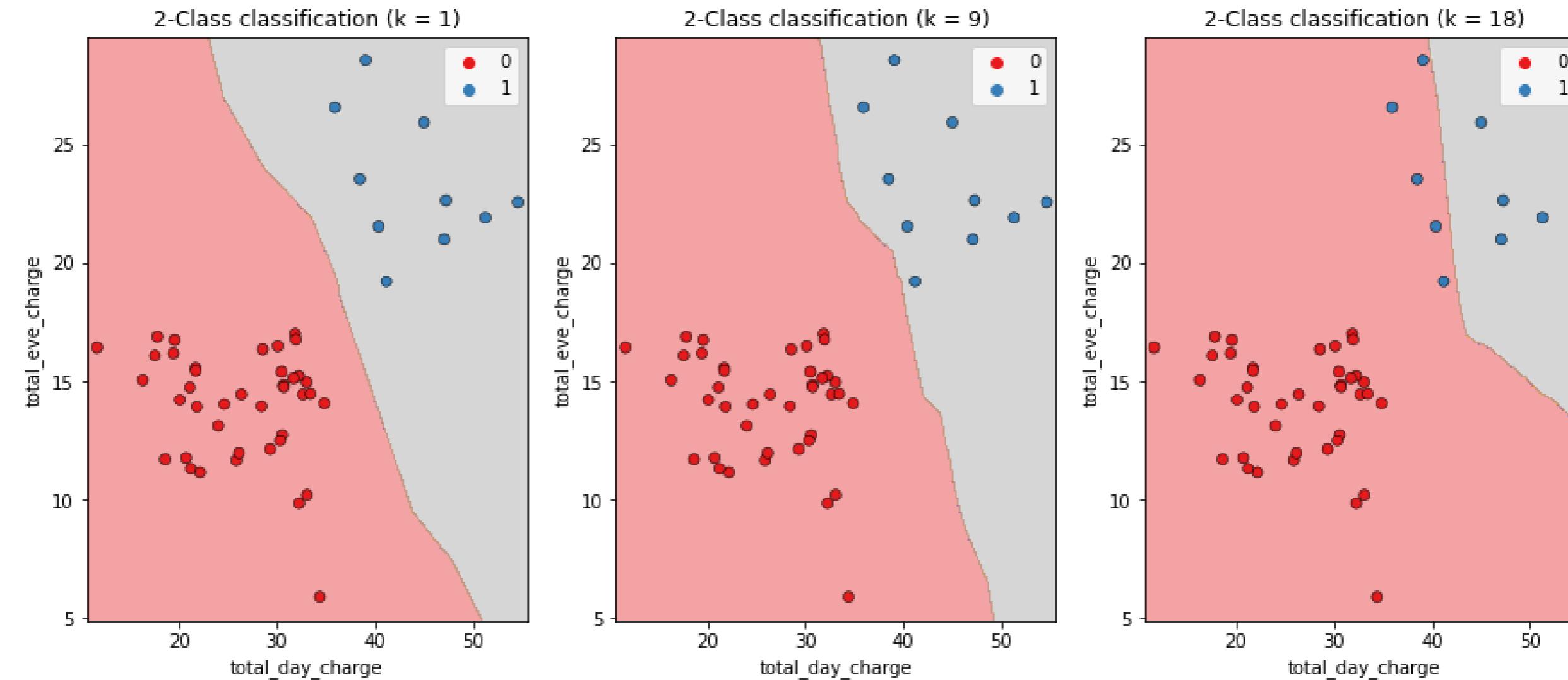
Train/test split

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=21, stratify=y)
knn = KNeighborsClassifier(n_neighbors=6)
knn.fit(X_train, y_train)
print(knn.score(X_test, y_test))
```

0.8800599700149925

Model complexity

- Larger k = less complex model = can cause underfitting
- Smaller k = more complex model = can lead to overfitting



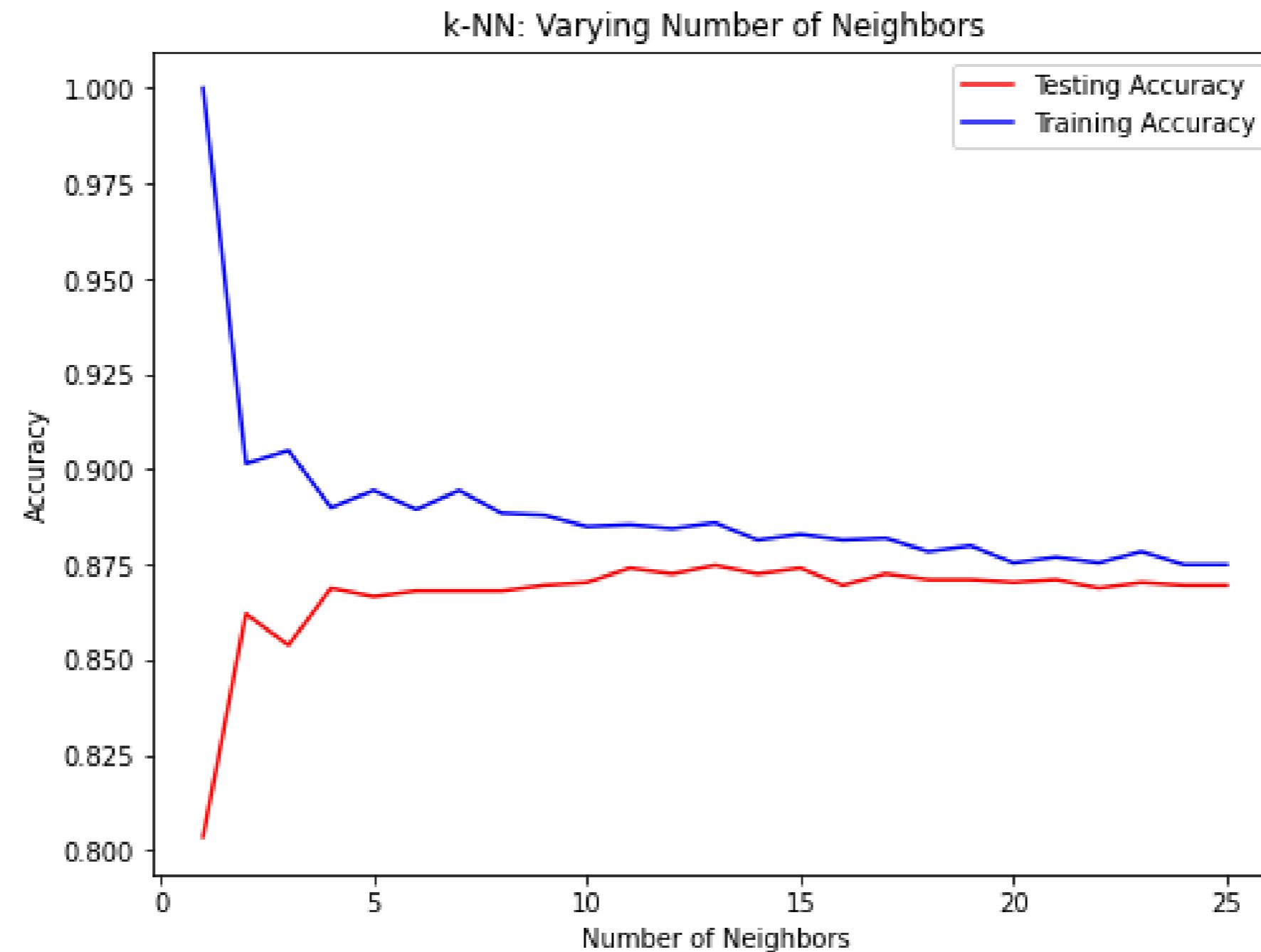
Model complexity and over/underfitting

```
train_accuracies = {}
test_accuracies = {}
neighbors = np.arange(1, 26)
for neighbor in neighbors:
    knn = KNeighborsClassifier(n_neighbors=neighbor)
    knn.fit(X_train, y_train)
    train_accuracies[neighbor] = knn.score(X_train, y_train)
    test_accuracies[neighbor] = knn.score(X_test, y_test)
```

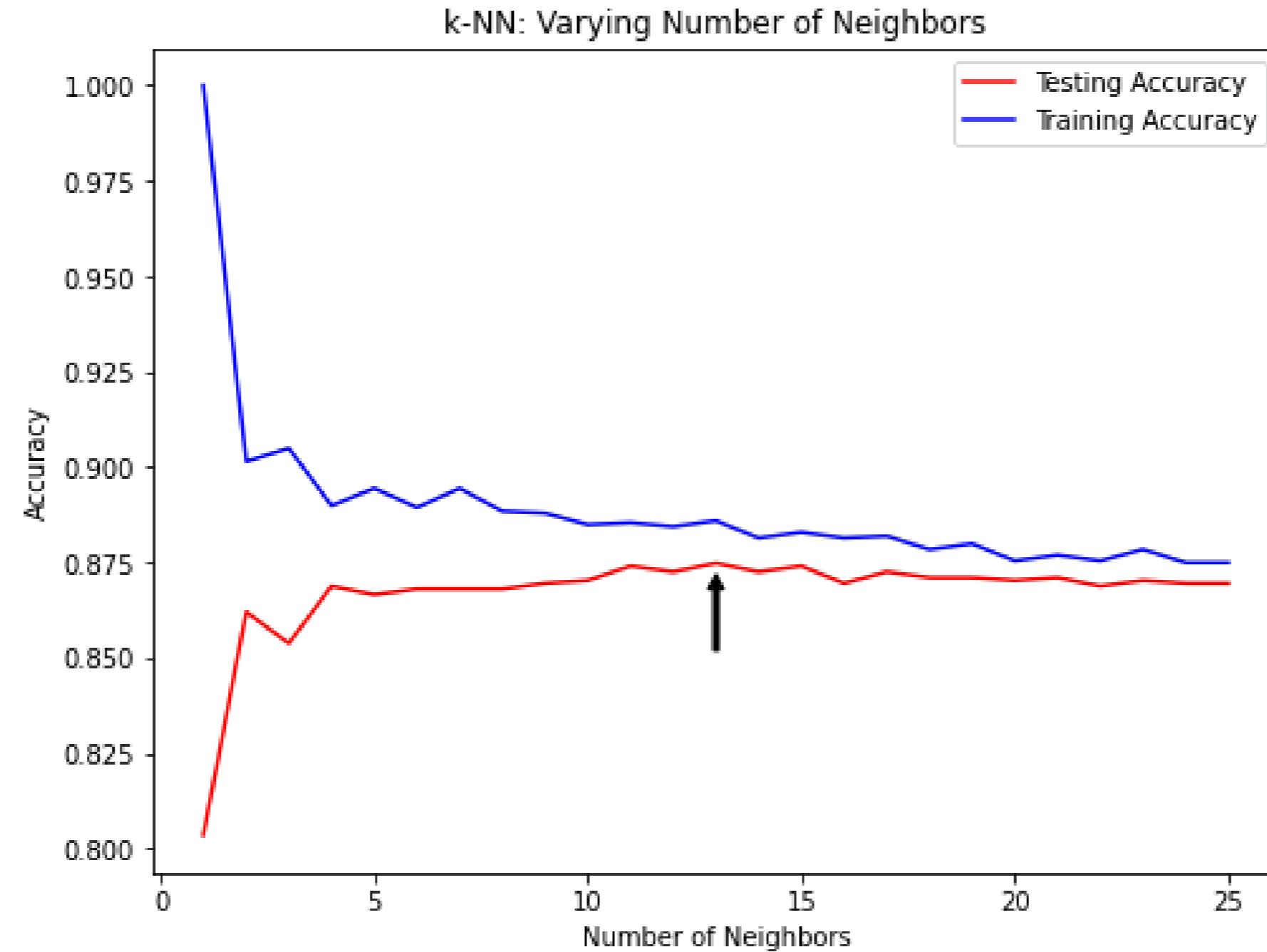
Plotting our results

```
plt.figure(figsize=(8, 6))
plt.title("KNN: Varying Number of Neighbors")
plt.plot(neighbors, train_accuracies.values(), label="Training Accuracy")
plt.plot(neighbors, test_accuracies.values(), label="Testing Accuracy")
plt.legend()
plt.xlabel("Number of Neighbors")
plt.ylabel("Accuracy")
plt.show()
```

Model complexity curve



Model complexity curve



Let's practice!

SUPERVISED LEARNING WITH SCIKIT-LEARN