K-NN Algorithm

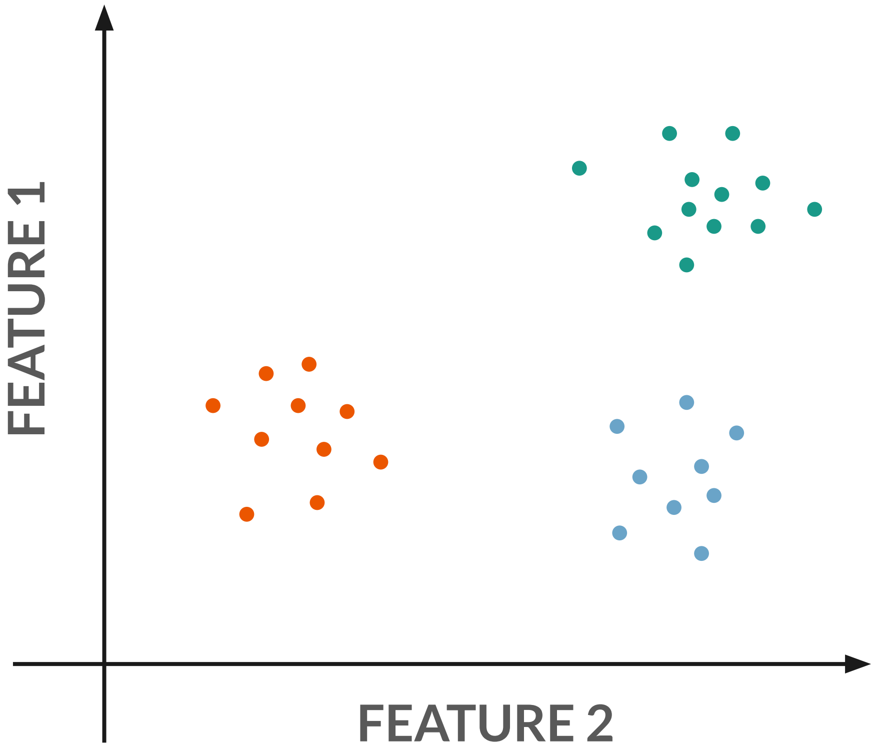
Regression & Classification

February 27th, 2024

Presented by Layth Al-Nemri, Pattarawat Dhanespramoj, Christopher Fitzgibbon, Neil Sanjiv Punjani, Ziying Xu

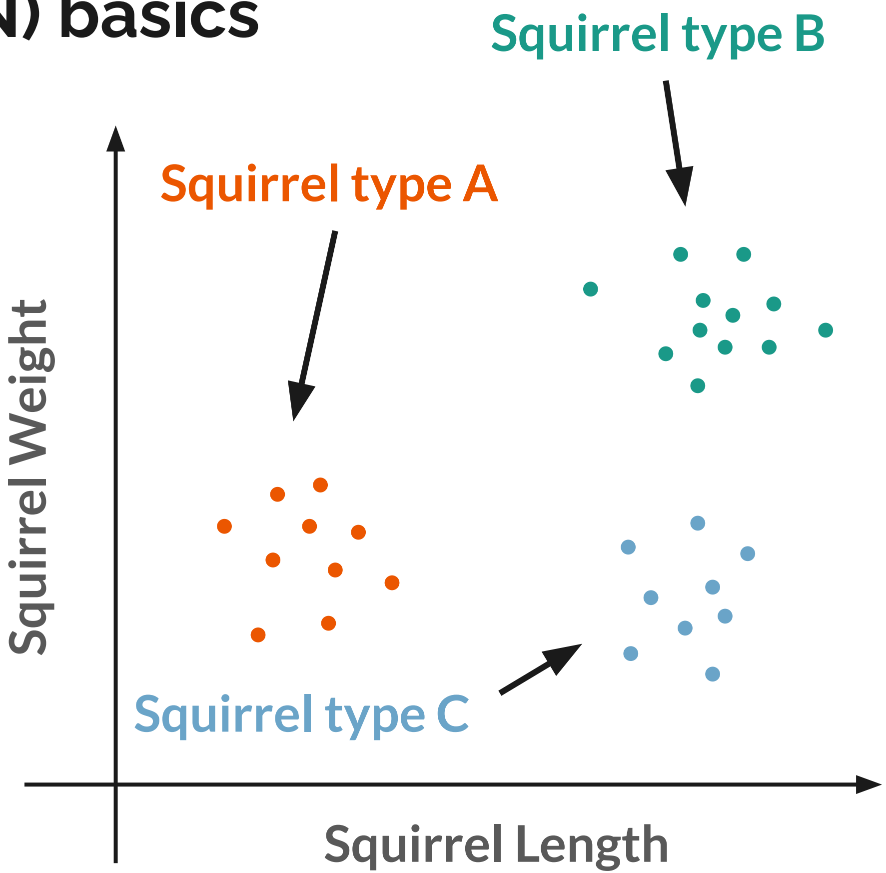
K-nearest neighbours (K-NN) basics

Suppose we have some data.



K-nearest neighbours (K-NN) basics

Perhaps we have WEIGHT plotted against LENGTH for some known squirrel types.

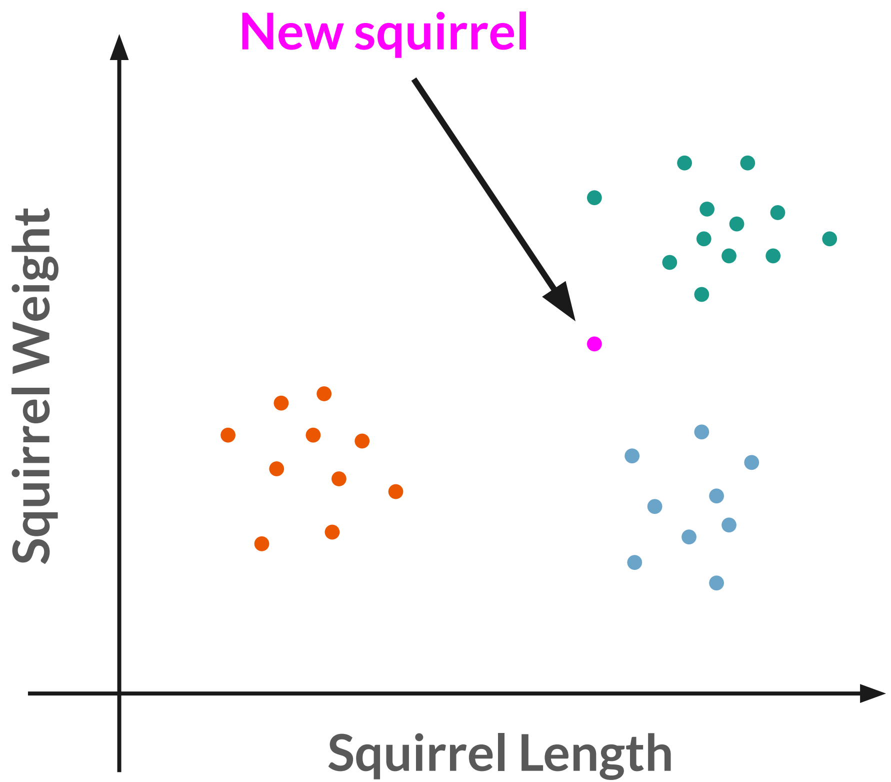


Now suppose we caught a squirrel and measured its weight and length.
With this data we can figure out what <u>type</u> of squirrel it is!

This is called **classification**.

K-NN classification

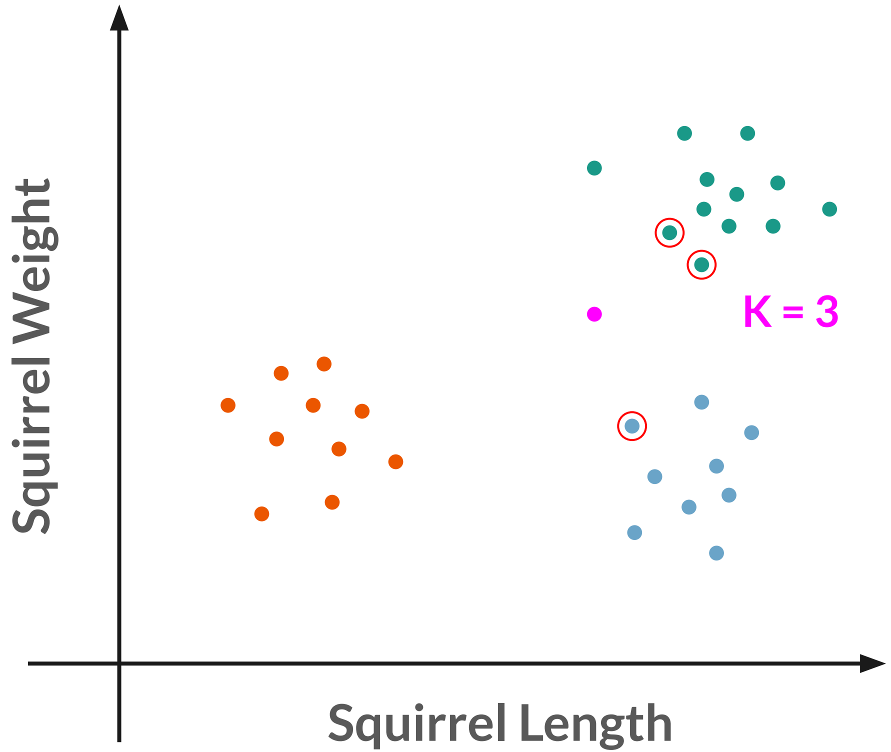
How can we use K-NN to figure out what type of squirrel this is?



K-NN classification

Step 1: Determine how many neighbours to assess.

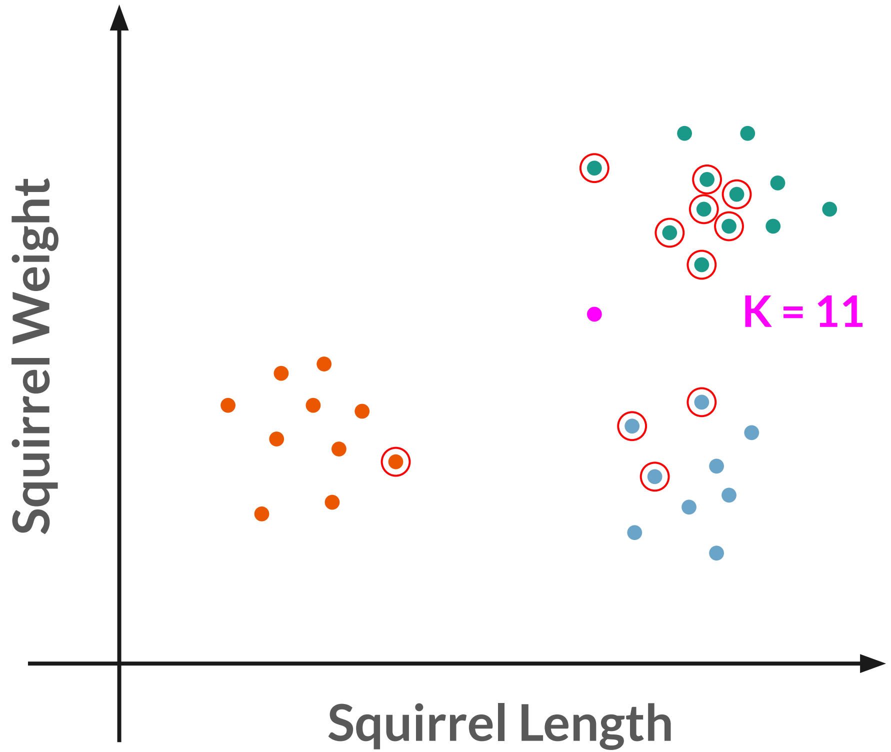
This is the k-value, and is the key hyperparameter for this algorithm.



K-NN classification

Step 1: Determine how many neighbours to assess.

This is the k-value, and is the key hyperparameter for this algorithm.

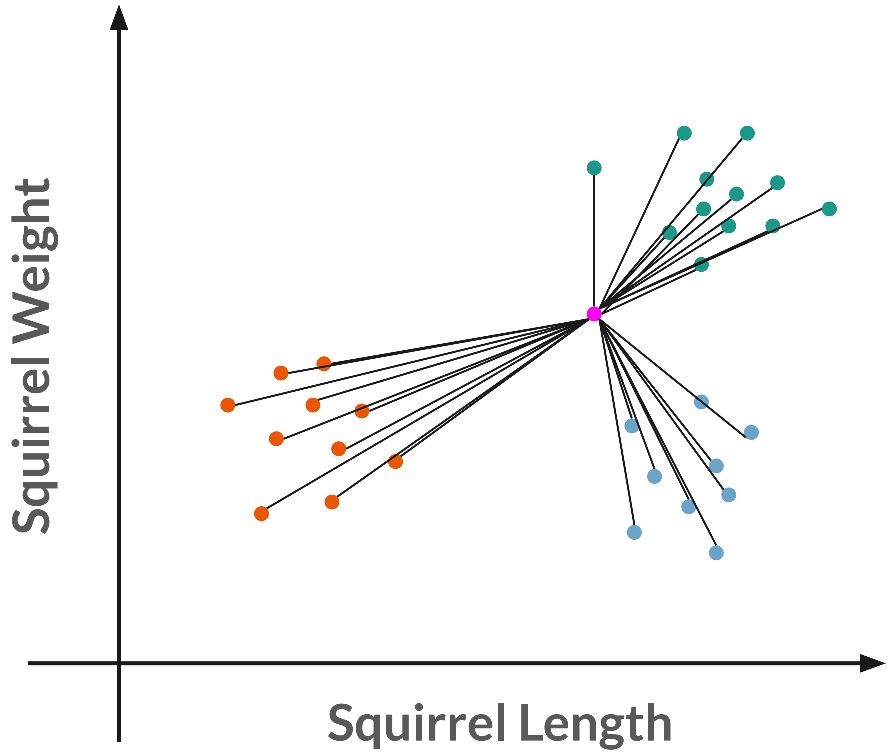


K-NN classification

Step 2: Calculate distance to all other points.

This is called **brute force**, and yes it's a bit computationally heavy.

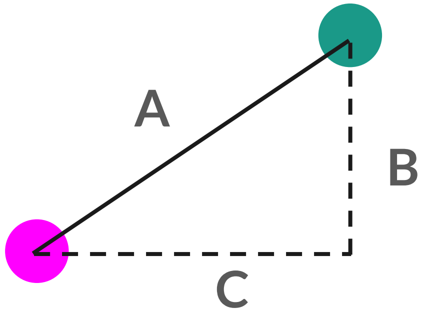
Alternatively algorithms **K-D tree** and **ball tree** have been developed for large datasets.



How is distance calculated?

Euclidean Distance

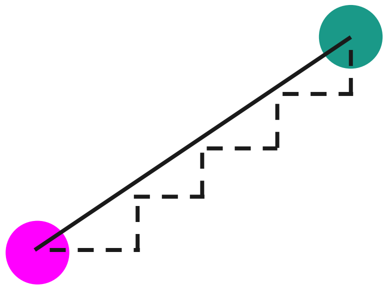
$$d(x, y) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$



(Typically default method)

Manhattan Distance

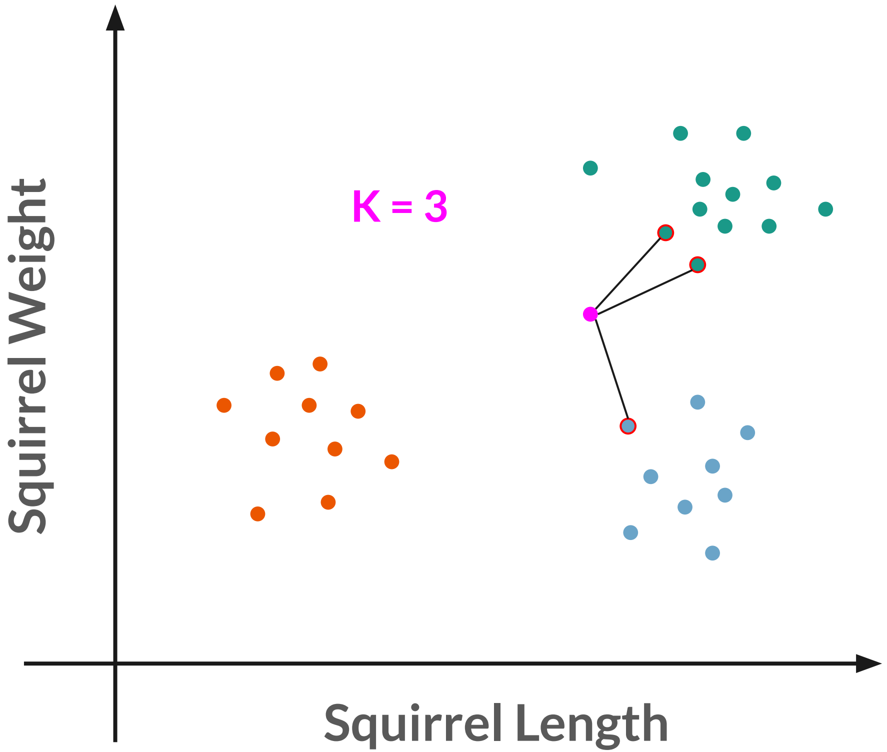
$$d(x, y) = \sum_{i=1}^{n} |x_i - y_i|$$



(Best for high dimensionality)

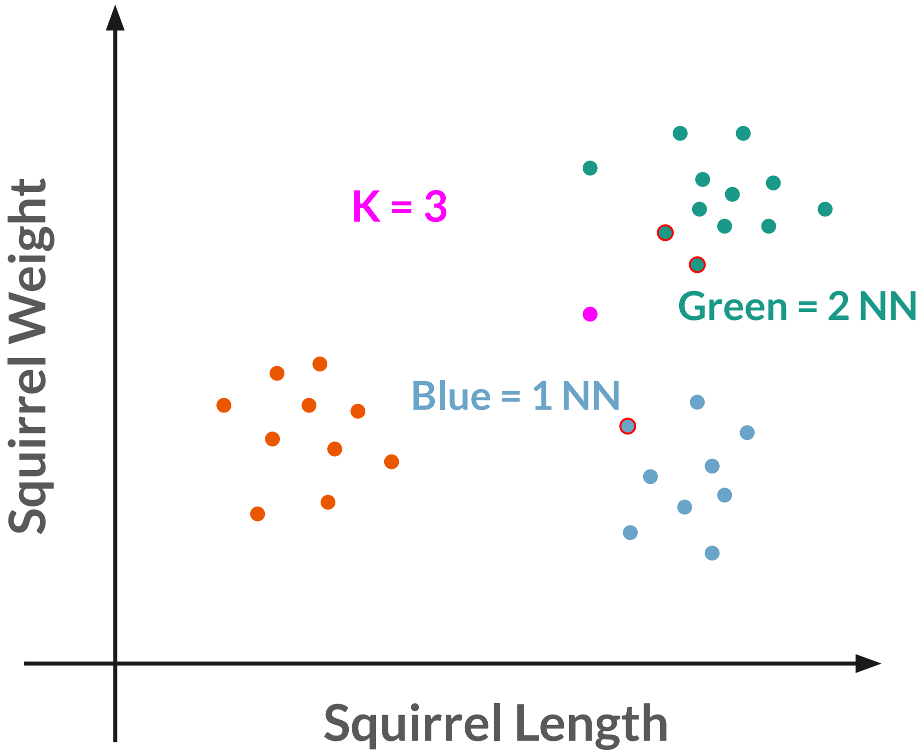
K-NN classification

Step 3: Filter for only k-nearest neighbors.



K-NN classification

Step 4: Classify the new squirrel based on which group had the <u>most</u> nearest neighbours.

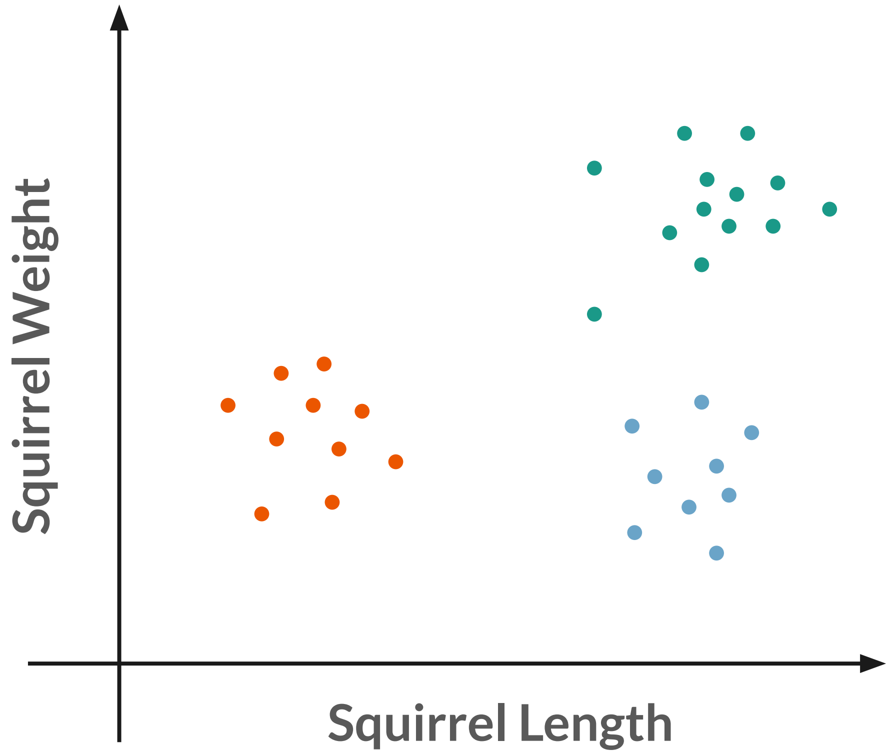


K-NN classification

Green wins!

The new squirrel is type B.

Performance usually assessed using **Accuracy.**



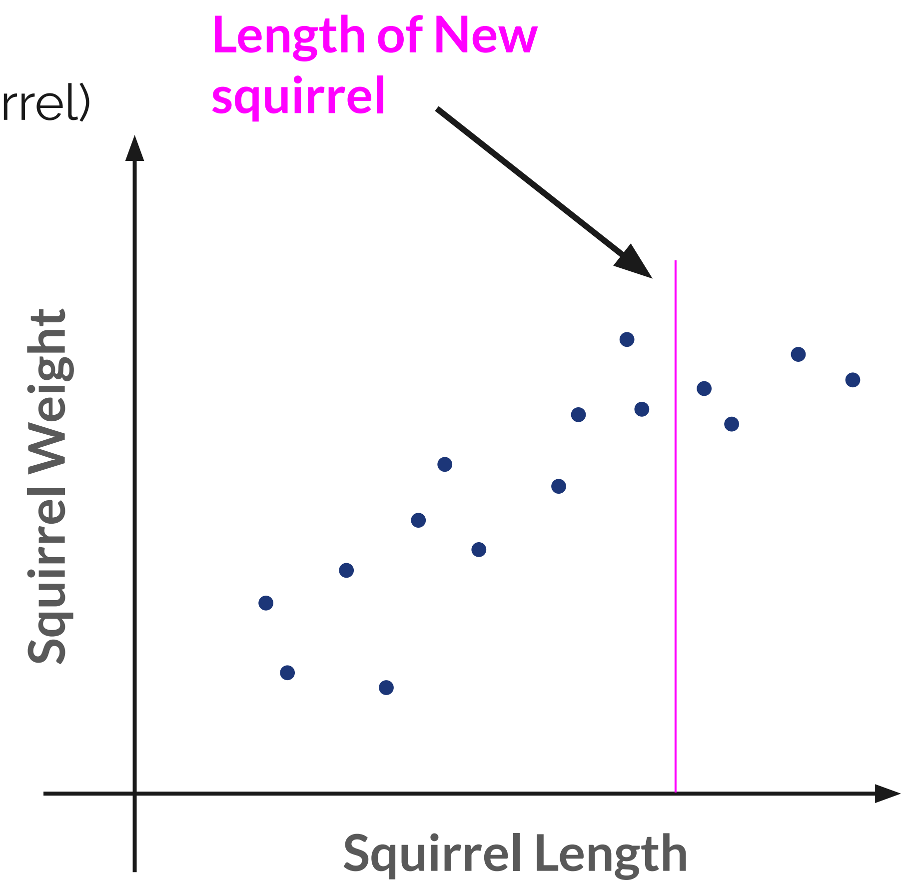
Now, suppose saw a squirrel at the zoo and managed to measure its length. We can use the k-NN method to predict its weight!

This is called **regression**.

K-NN regression

(new dataset for a certain type of squirrel)

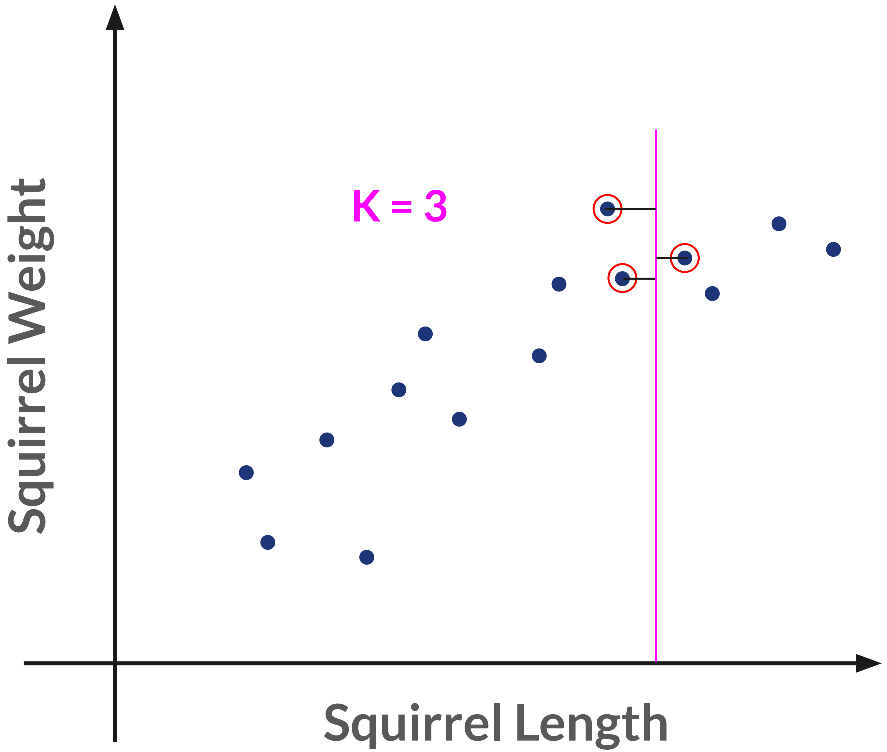
How can we use K-NN to predict the weight of a Type C squirrel?



K-NN regression

Same as before,

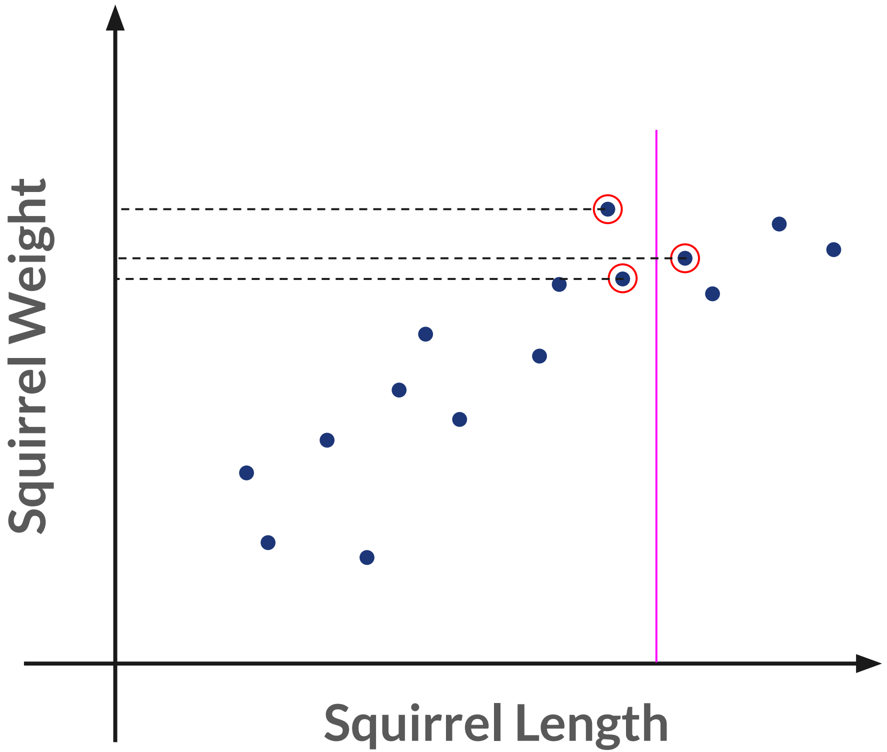
1. Set k (let's say k = 3)
2. Calculate distance to find NNs



K-NN regression

Now,

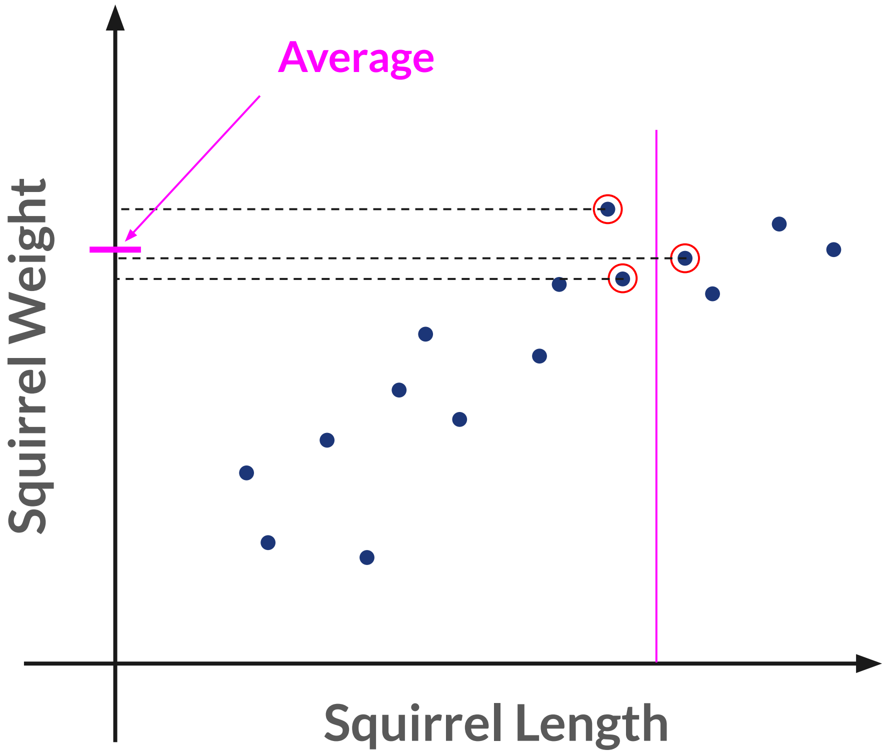
3. Determine y-values for the NNs



K-NN regression

Now,

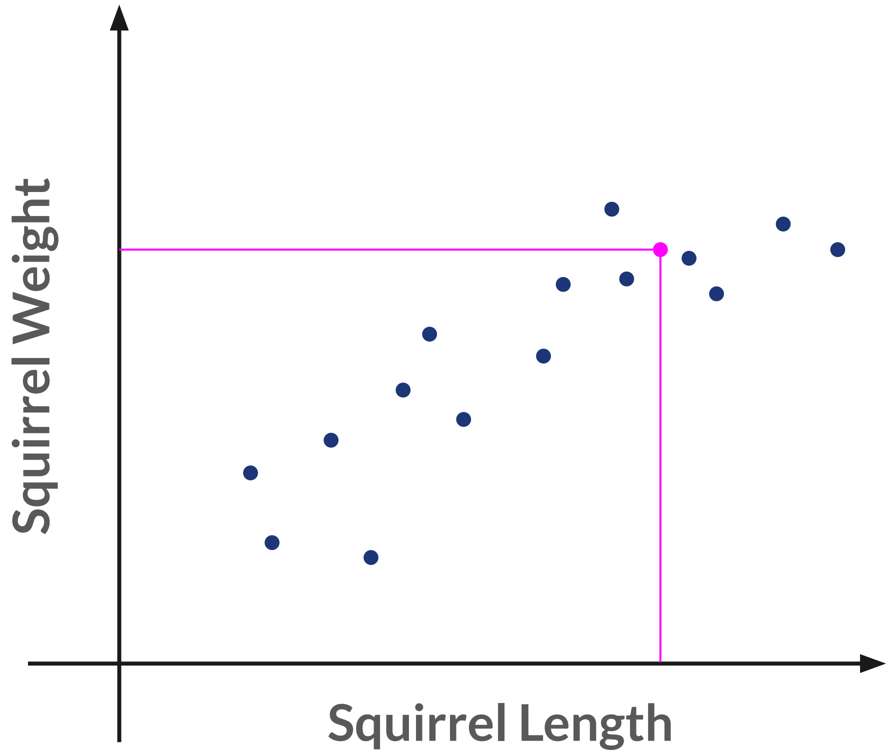
3. Determine y-values for the NNs
4. Calculate average



K-NN regression

The squirrel weight is calculated!

Performance usually assessed using **MSE.**



k-NN Algorithms in Practice

Our Classification and Regression Problems

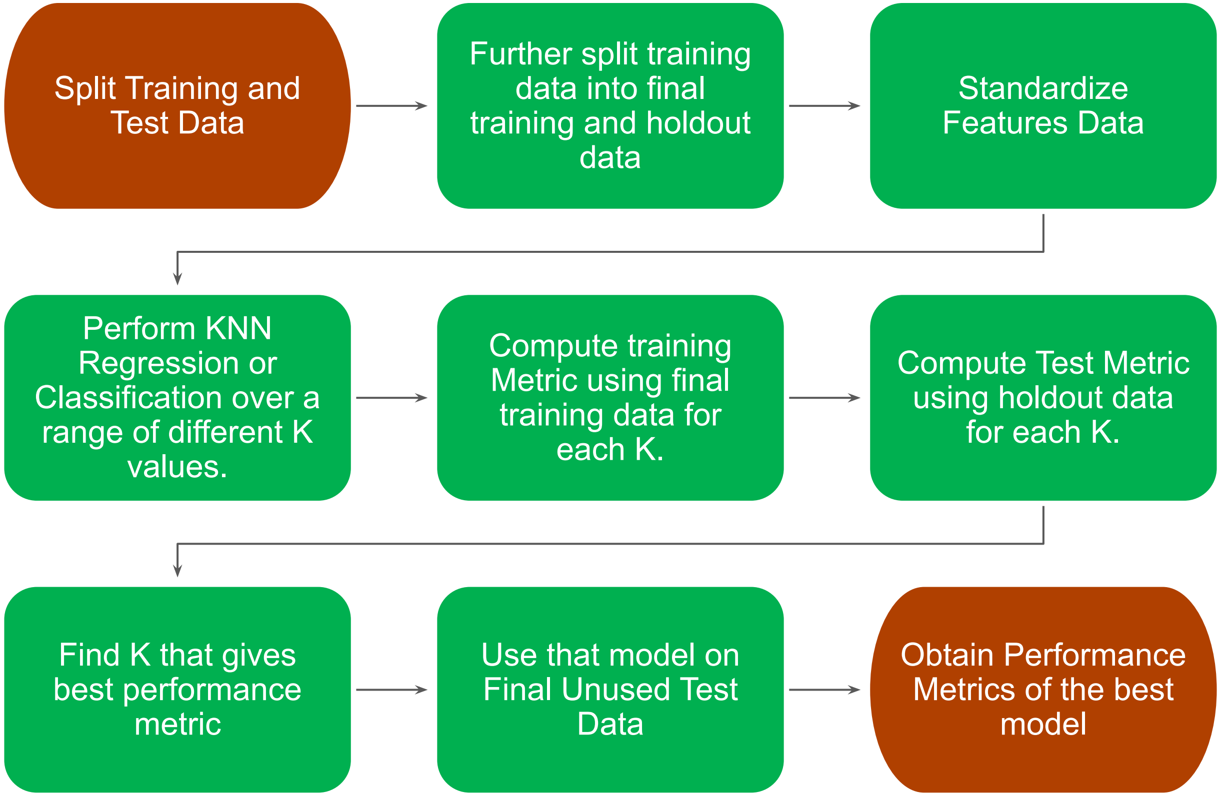
Classification - Predicting the quality of wine (target) using different wine characteristics (features)

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality	Id
0	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	0
1	7.8	0.88	0.00	2.6	0.098	25.0	67.0	0.9968	3.20	0.68	9.8	5	1
2	7.8	0.76	0.04	2.3	0.092	15.0	54.0	0.9970	3.26	0.65	9.8	5	2
3	11.2	0.28	0.56	1.9	0.075	17.0	60.0	0.9980	3.16	0.58	9.8	6	3
4	7.4	0.70	0.00	1.9	0.076	11.0	34.0	0.9978	3.51	0.56	9.4	5	4

Regression - Predicting the median house price (target) using different housing characteristics (features)

	longitude	latitude	housing_median_age	total_rooms	total_bedrooms	population	households	median_income	median_house_value
0	-122.23	37.88	41.0	880.0	129.0	322.0	126.0	8.3252	452600.0
1	-122.22	37.86	21.0	7099.0	1106.0	2401.0	1138.0	8.3014	358500.0
2	-122.24	37.85	52.0	1467.0	190.0	496.0	177.0	7.2574	352100.0
3	-122.25	37.85	52.0	1274.0	235.0	558.0	219.0	5.6431	341300.0
4	-122.25	37.85	52.0	1627.0	280.0	565.0	259.0	3.8462	342200.0

KNN classification and regression using holdout data



Holdout Method Code for Classification (Regression)

Step 1:

```
X_train_total, X_test, y_train_total, y_test = train_test_split(features, targets, test_size=0.2, random_state=0)
X_train, X_val, y_train, y_val = train_test_split(X_train_total, y_train_total, test_size=0.2, random_state=0)
```

Split Data

Step 2:

```
scaler = preprocessing.StandardScaler().fit(X_train_total)#fitting the cross validation training set for standardization
X_train_total_scaled = scaler.transform(X_train_total)#standardizing the cross validation training set
X_test_total_scaled = scaler.transform(X_test)#standardizing the test set
```

Standardize the data

Step 3:

```
for k in k_values:
 knn_b = KNeighborsClassifier(n_neighbors=k)
 knn_b.fit(X_train_scaled, y_train)
 y_pred_2 = knn_b.predict(X_val_scaled)
 y_pred_2_t = knn_b.predict(X_train_scaled)#getting predictions for the training set
 current_accuracy = accuracy_score(y_val, y_pred_2)
 accuracy_t = accuracy_score(y_train, y_pred_2_t)
 accuracy_no_cv.append(current_accuracy)
 accuracy_no_cv_t.append(accuracy_t)
 if best_accuracy_no_cv < current_accuracy:
   best_accuracy_no_cv = current_accuracy
   best_k_no_cv = k
```

```
#iterate through hyperparameters
for k in k_values:
  #declare and fit the model
  neigh_a = neighbors.KNeighborsRegressor(n_neighbors = k)
  neigh_a.fit(X_train, y_train)
  #make prediction
  prediction_val_noSC = neigh_a.predict(X_val)
  prediction_train_noSC = neigh_a.predict(X_train)
```

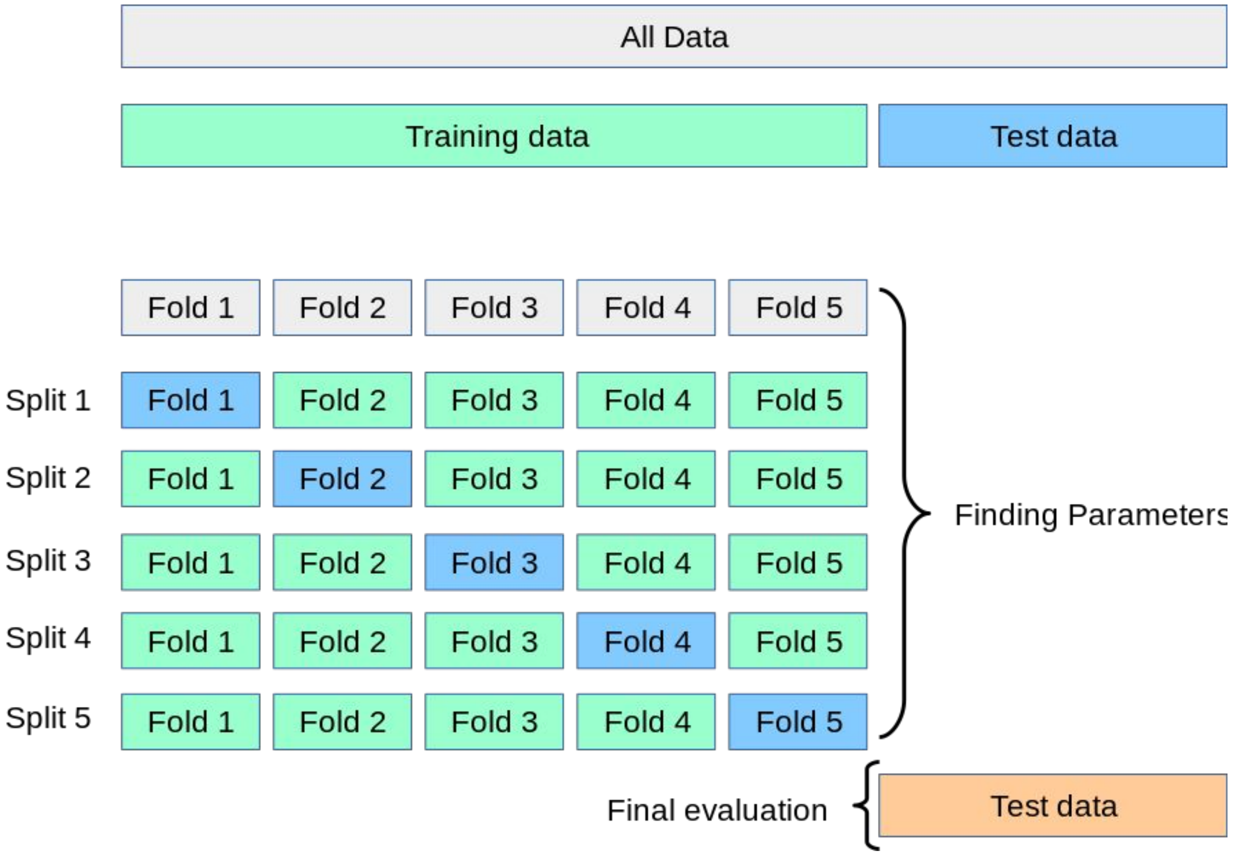
Fit data and find the best k

Step 4:

```
#training a model using k obtained with standardization and cross validation and testing it
knn_e = KNeighborsClassifier(n_neighbors=best_k)
knn_e.fit(X_train_total_scaled, y_train_total)
y_pred_3 = knn_e.predict(X_test_total_scaled)
accuracy = accuracy_score(y_test, y_pred_3
```

Fit Model on unseen data

KNN classification and regression using Cross Validation



Cross Validation Method Code for Classification and Regression

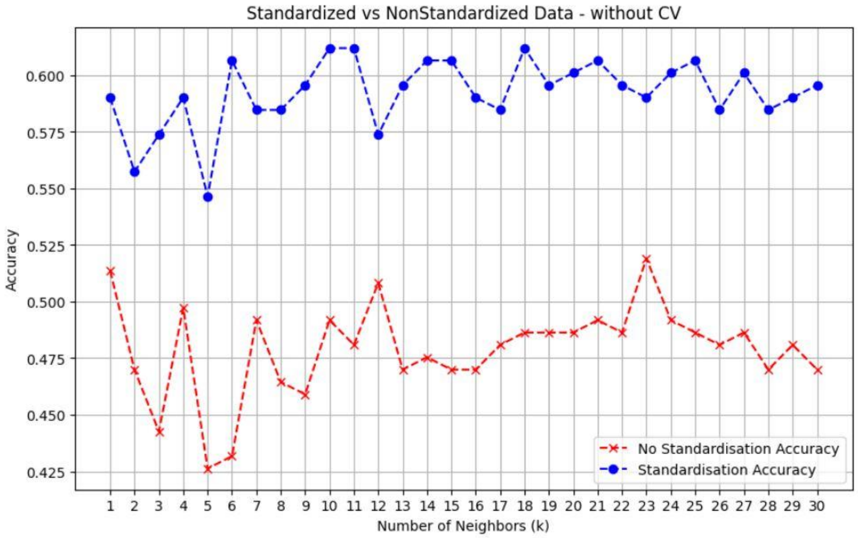
```
Step 1:
X_train_total, X_test, y_train_total, y_test = train_test_split(features, targets, test_size=0.2, random_state=0) #splitting the data into training and test sets
X_train, X_val, y_train, y_val = train_test_split(X_train_total, y_train_total, test_size=0.2, random_state=0) #splitting the training set into training and validation

Step 2:
scaler = preprocessing.StandardScaler().fit(X_train_total)#fitting the cross validation training set for standardization
X_train_total_scaled = scaler.transform(X_train_total)#standardizing the cross validation training set
X_test_total_scaled = scaler.transform(X_test)#standardizing the test set

Step 3:
for k in k_values:
 knn_c = KNeighborsClassifier(n_neighbors=k)
 scores = cross_validate(knn_c, X_train_total_scaled, y_train_total, cv=5, scoring='accuracy', return_train_score=True) #training the knn model through cross validation and storing the scores
 cv_current_score = scores['test_score'].mean() #storing the validation scores
 cv_current_score_t = scores['train_score'].mean() #scoring the training scores
 cv_scores.append(cv_current_score)
 cv_scores_t.append(cv_current_score_t)
 if best_accuracy < cv_current_score:
   best_accuracy = cv_current_score
   best_k = k

Step 4:
#training a model using k obtained with standardization and cross validation and testing it
knn_e = KNeighborsClassifier(n_neighbors=best_k)
knn_e.fit(X_train_total_scaled, y_train_total)
y_pred_3 = knn_e.predict(X_test_total_scaled)
accuracy = accuracy_score(y_test, y_pred_3)
```

Importance of Standardization

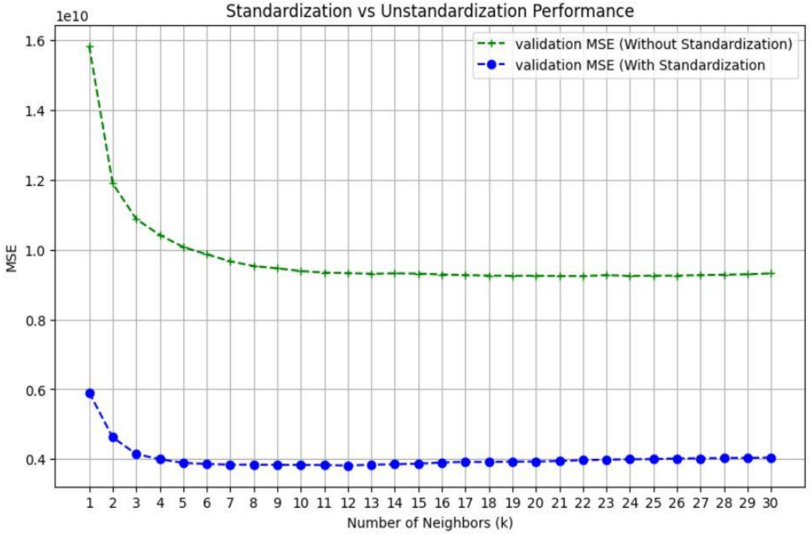
Algorithm will prioritize features with higher numerical values!

Impact of other features will get overshadowed! Potentially Poorer Performance.

$$z = \frac{x_i - \mu}{\sigma}$$



Test Accuracy for Classification - Hold out method - (Wine Data)



Test Accuracy for Regression – Hold Out method - (Housing Data)

Bias-Variance Tradeoff Overview for KNN

A low k value may cause training data to overfit. – Performs worse on Unseen Data

Low Bias ↓ **High Variance** ↑

A high k value may cause training data to underfit. – Does not capture model complexity

High Bias ↑ **Low Variance** ↓

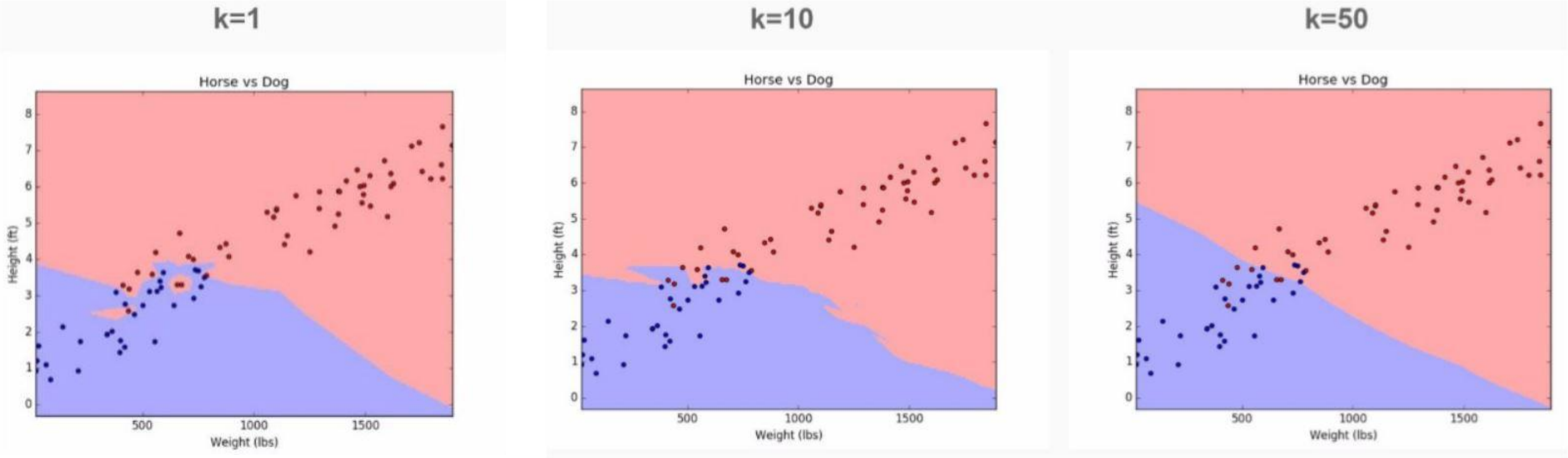
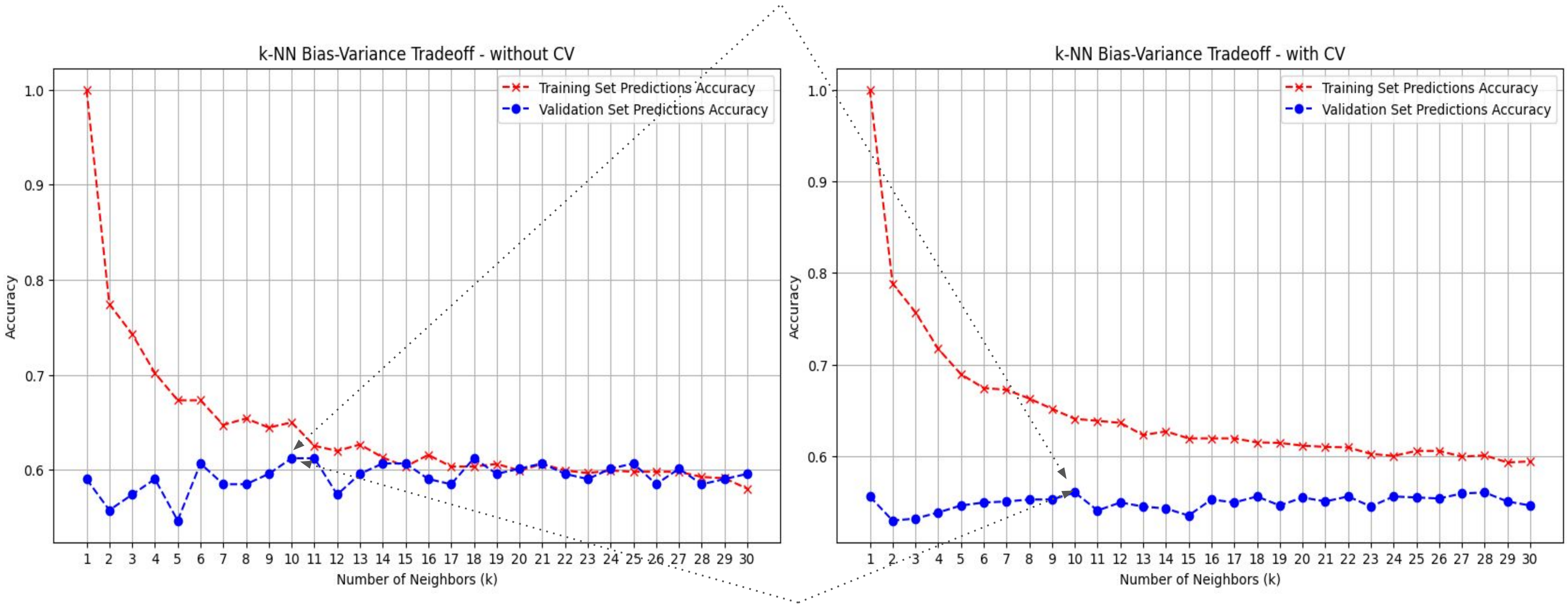


Image Source: https://datascienceplus.com/k-nearest-neighbors-knn-with-python/

Bias-Variance Tradeoff for our Classification Model

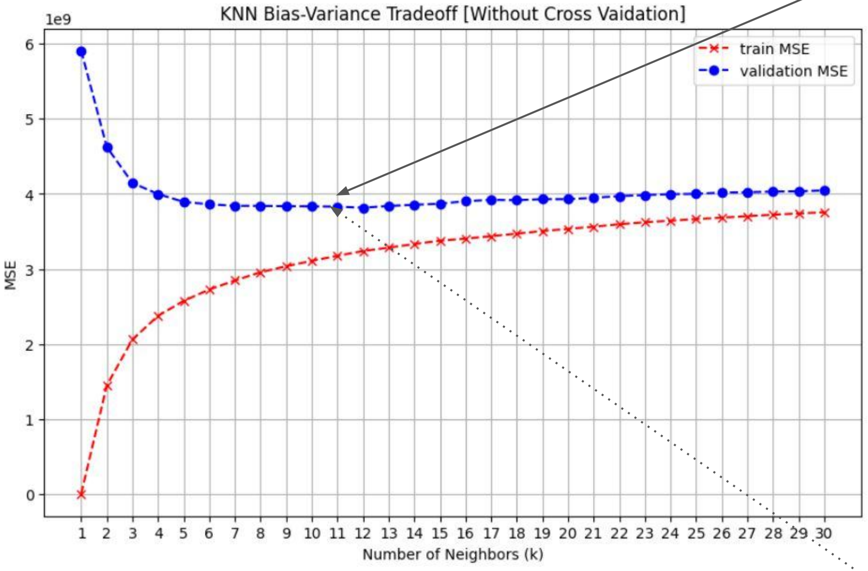
Best Validation Performance - Ideal Bias – Variance Tradeoff Point

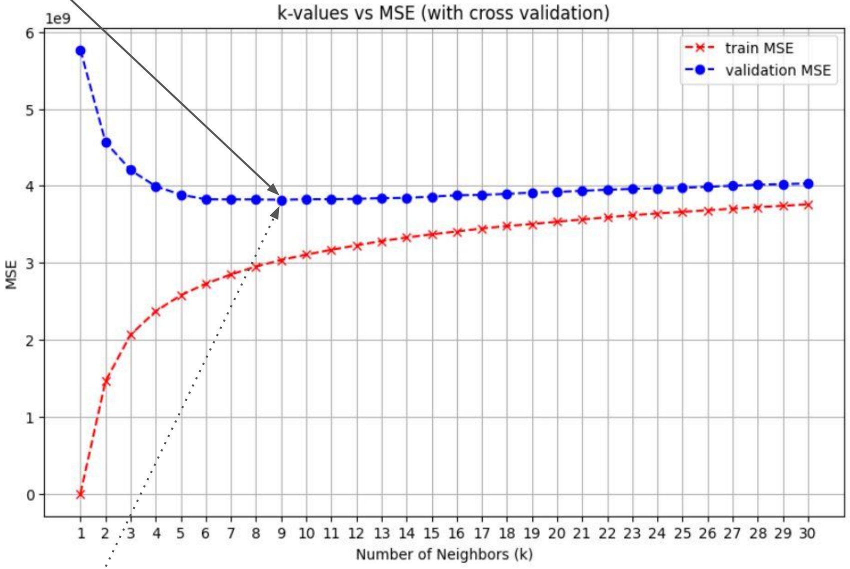


Hyperparameter Tuning – Best Test Set Performance (k)

Bias-Variance Tradeoff for our Regression Model

Best Validation Performance - Ideal Bias – Variance Tradeoff Point





Hyperparameter Tuning – Best Validation Set Performance (k)

Test Results for our Models

Classification Final Test Performance

	Holdout Accuracy	Cross Validation Accuracy
Classification	**0.5939**	0.5895

Higher accuracy

Lower error

Regression Final Test Performance

	Holdout MSE	Cross Validation MSE
Regression	3882647924.3960	**3839934272.6308**

Holdout method performed superior in Classification and CV for Regression Problem for our final test.

Cross Validation or Holdout method?

How do we evaluate which method to choose for our model tuning process?

	Holdout Method	Cross Validation
Pros	Faster Computation Time Simpler to implement	Useful for smaller datasets. Uses entire training set for testing. Provides more reliable model performance metric.
Cons	Model may have high variance Biased estimates if data split not representative	Complex to implement Requires more computational resources

CV offers more robust model to different unseen test data!

Holdout performing better on classification probably due to split and test set characteristics.

References

- https://www.kaggle.com/datasets/yasserh/wine-quality-dataset/code - Wine Quality Dataset
- https://www.kaggle.com/datasets/camnugent/california-housing-prices?resource=download - California Housing Dataset
- https://towardsmachinelearning.blogspot.com/2023/02/what-is-k-fold-cross-validation-working.html - CV Image
- ChatGPT for assisting in syntax