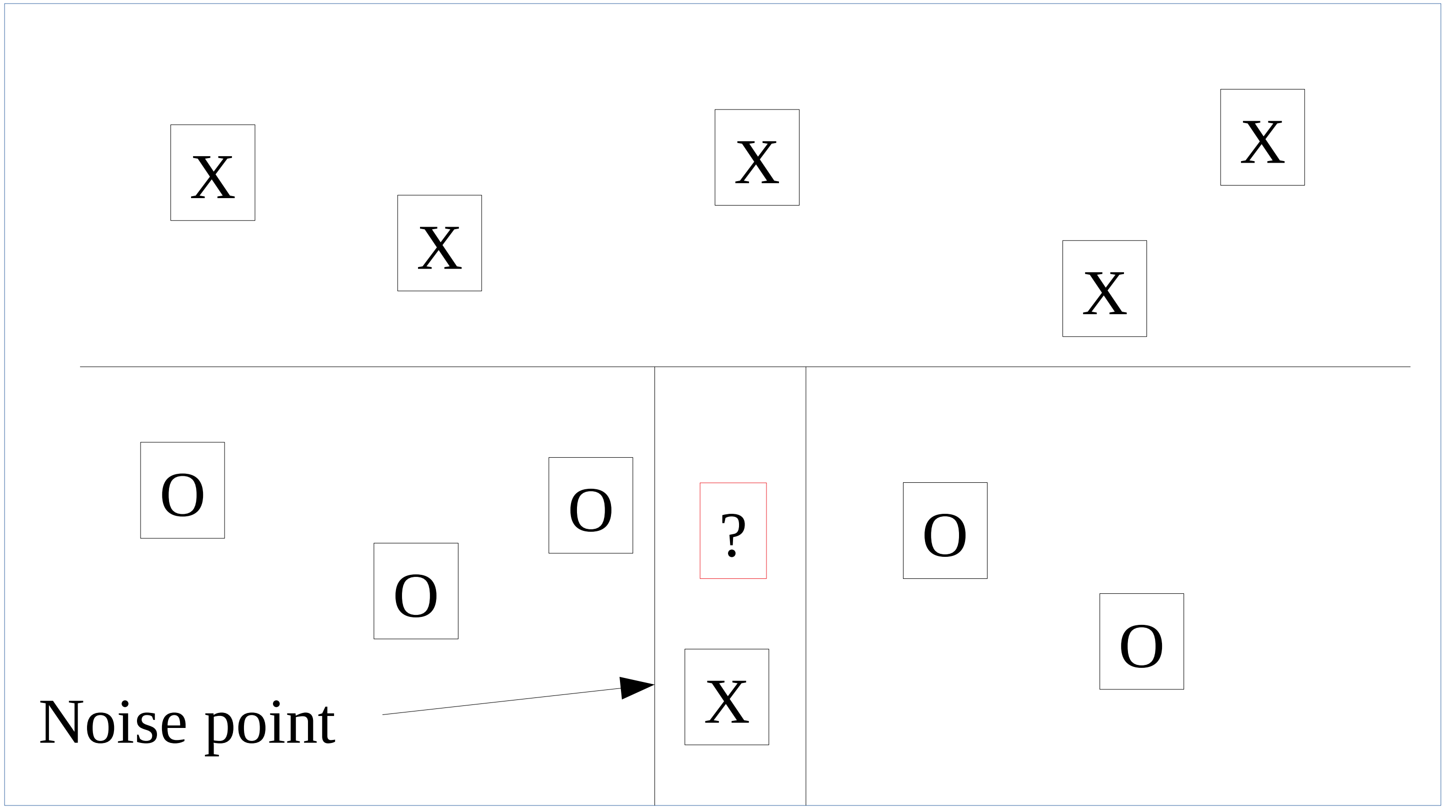
Decision Trees

Overfitting and Cross Validation

Decision Tree Algorithm

- For each feature, determine all the possible splits for that feature and use node impurity measure to choose the best split for this feature.
- Compare all these best splits and so choose the best feature to split.
- Implement the split, that is create child nodes in the decision tree for the split.
- Start with the root node and recursively do this until a stop condition is reached.

Overfitting



Overfitting

- A noise point is an outlier in the data due to for example an incorrect measurement, or some other unknown influence.
- It is wrong to try and model every point in the training data expecially if contains noise points.
- For example is the red point above an x or an o.
- If x is a noise point then the red point is probably an o.

Overfitting with Decision Trees

- Decision trees are a powerful modelling technique.
- It is often possible to continually split the data until every instance in the training set is classified correctly.
- This normally leads to overfitting.
- And the decision tree will normally perform less well on unseen data.

Overfitting with Decision Trees

- Overfitting results in decision trees that are more complex than necessary
- Training error no longer provides a good estimate of how well the tree will perform on previously unseen records

Early Stopping

- Decision tree will stop splitting nodes when
  - All instances in a node are in the same class (normally)
  - Or all instances have the same attribute values
- Early stopping
  - Stop if the number of instances in a node is smaller that a “bucket size”
  - Stop if the GINI value doesn’t decrease.
  - Stop if the tree gets too deep.

Meta-Parameters

- Bucket size, or depth of a decision tree is an example of a meta parameter.
- These values can not be determined by the training data, only by applying the resultant model to unseen data.
- In order to determine a value for a meta-parameter, validation data is used.

Validation Data

- So a data set is divided into
  - Training data
  - Validation data
  - Test data
- By separating the validation data from the test data, the test data is a more accurate estimate of the accuracy of the model as the test data has not been involved in building the model.

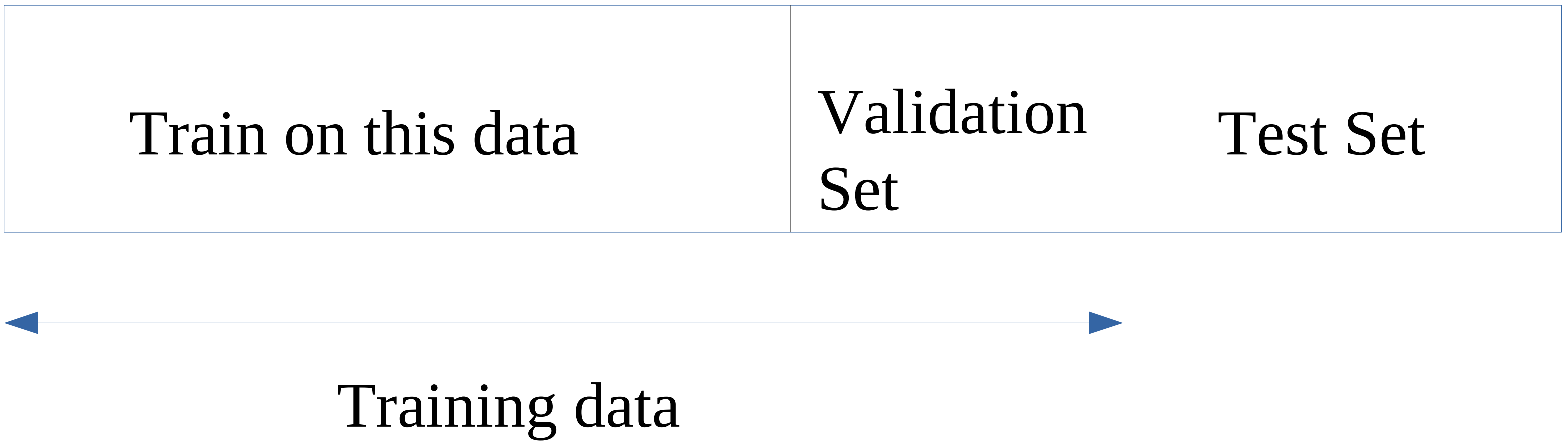
Depth of Decision Tree

- If a decision tree is too large then the splits at the bottom of the tree are dealing wih very small number of instances.
- These splits can cause overfitting, that is they work for the traing set but don’t generalize to previously unseen data.
- We are going to controll overfitting by controlling the dept of the decision tree.

Validation Set

- The test data set should never be used when picking a model.
- The validation set is used to pick the model.

Validation Set



Train on this data

Validation Set

Test Set

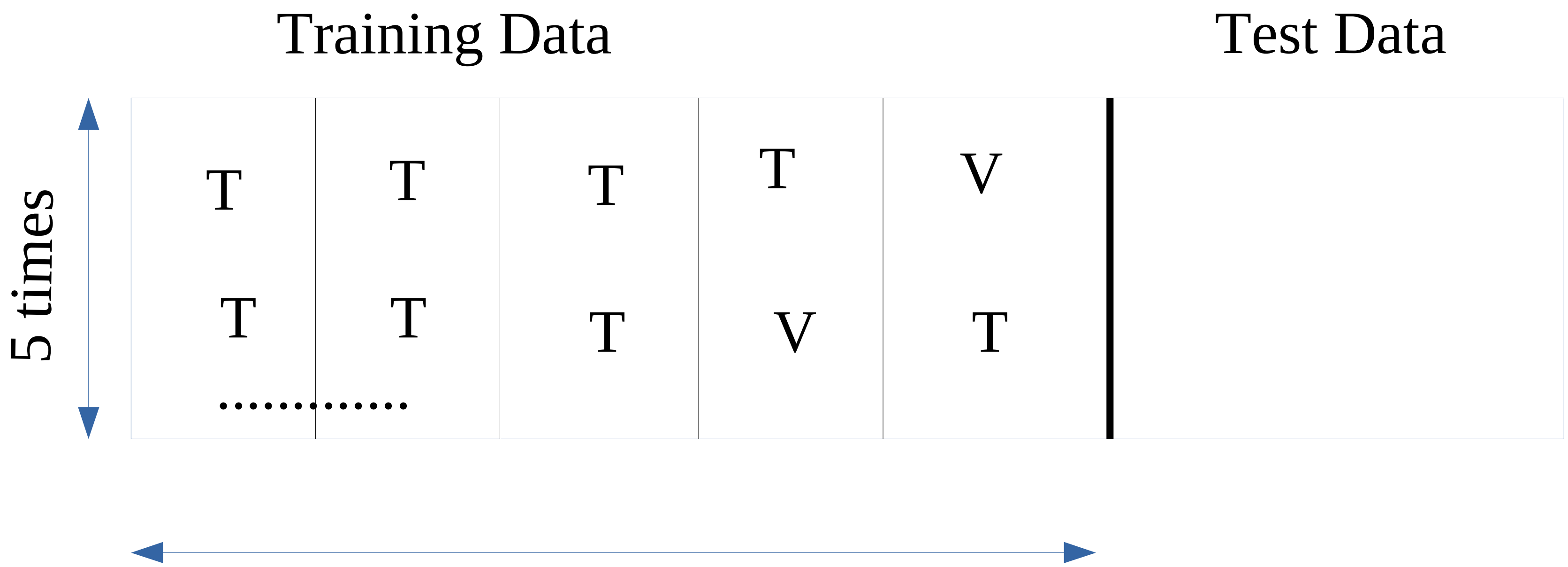
Training data

- Train different models using the training data.
- Pick the best model using the validation data.
- Evaluate the model using the test data.

Cross Validation

- In practise, in order to get the most out of the training data, cross validation is used.
- The original training data is divided into 5 (n).
- The model is trained on 4 parts and (for the purpose of picking the best model) evaluated on 1 part.
- This is done 5 times with a different validation set each time.
- The average value of the validation accuracy is taken.

Cross Validation



Training Data

Test Data

5 times

T	T	T	T	V	
T	T	T	V	T	
…………					

5 fold cross valitation

cross_val_score

```
from sklearn.model_selection import cross_val_score
scores = cross_val_score(model, X_train, y_train, cv=5)
print(scores)
print(scores.mean())


[0.68518519 0.76851852 0.71028037 0.6728972  0.71962617]
0.7113014884042922
```

Cross Validation

- The mean of the 5 validation accuracies is a better estimate of model accuracy.
- Notice that only the training data is used to train and pick the model.
- This is because the test data has to be kept for evaluation ppurposes after the best value for the depth has been found.
-

(Final) Testing

- Cross validation is done for a number of different depths.
- The depth that gives the best average accuracy is chosen.
- Then the model with that depth is tested against the test data.

Cross Validation

```
for d in range(2,20) :
    tree = DecisionTreeClassifier(max_depth=d)

    scores = cross_val_score(tree, X_train, y_train, cv=5)
    print(scores.mean())
```

(Final) Testing

- Suppose 5 gives the best average accuracy.
- A Decision tree is created with max_depth=5.
- It is trained (fit) on the entire training data.
- And evaluated on the test data.