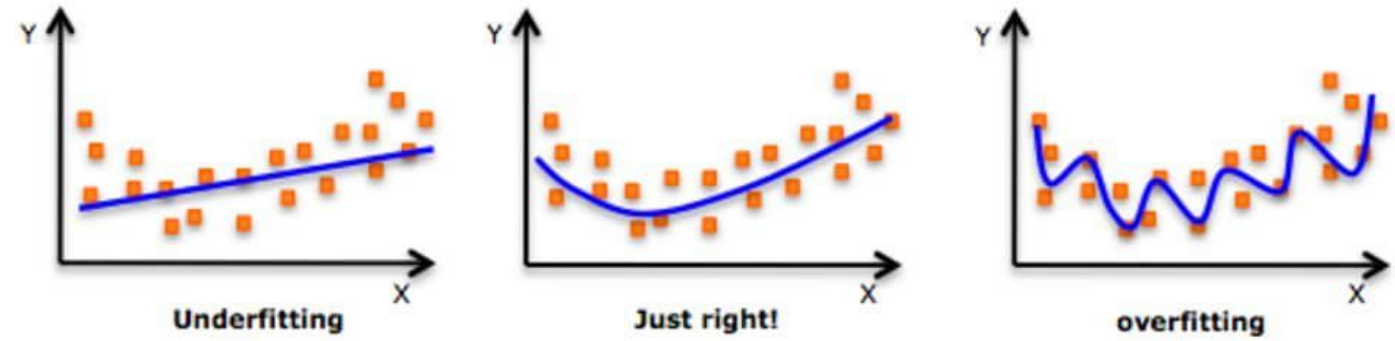
Model performance on train data

Model performance on test data

Swipe to learn + code

This problem is called overfitting



- One on the major problem while training ML problems is overfitting or underfitting.
- The model will have very high training accuracy and less test accuracy when the model is overfitted.
- When we try to map the function between input and output, then it tries to completely see the data and memorize everything whatever it had seen then it works great on train data and don't do well on test data.
- We should expect algorithm to have some variance. Ideally it should not change too much from one training dataset to next, meaning that algorithm is good at picking out the hidden patterns

Swipe <<<

Methods to avoid overfitting

Cross-validation

We keep one part of train data as validation data and remaining as train. To keep lower variance a higher fold cross validation is preferred.

Early stopping

It provide guidance as to how many iterations can be run before the learner begins to overfit.

Pruning

It is used extensively while building CART models. It simple removes the nodes which add little predictive power for the problem in hand.

Swipe <<<

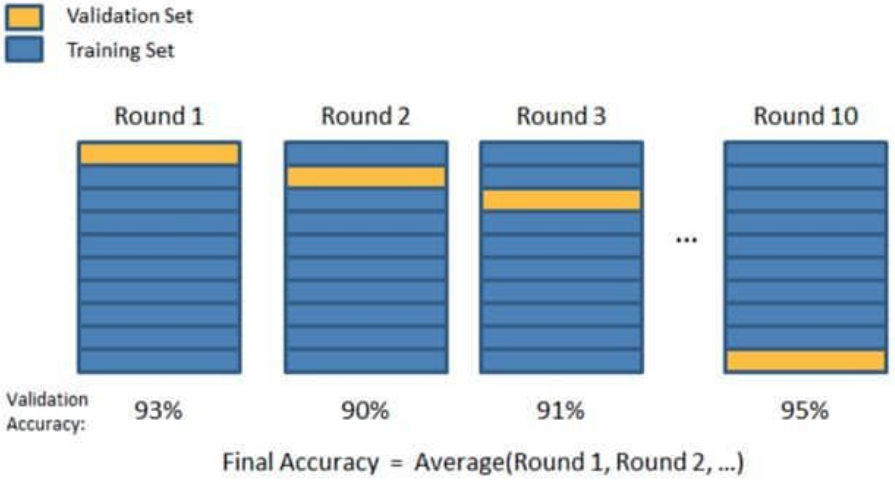
Methods to avoid overfitting

Regularization

It introduces a cost term for bringing in more features with the objective function. Hence , it tries to push the coefficients for many variables >= zero and reduce the cost term

swipe <<< for code

Cross validation



This is what cross validation looks like you set one part as validation and remaining part as train data. If fold = 10 then you will divide train data into 10 parts and you take each part as validation on every model you train and take the average of the result.

```
# NImport necessary libraries
from sklearn.model_selection import cross_val_score, cross_val_predict

# model = your model object, cv = no of folds
scores = cross_val_score(model, x_train, y_train, cv=10)
print(scores)

# make predictions
predictions = cross_val_predict(model, x_test, y_test, cv=10)
```

Swipe <<<

Regularization

```python
# NImport necessary libraries
from sklearn.linear_model import Ridge, Lasso

# Ridge model, alpha is a hyperparameter
ridgeModel = Ridge(alpha = 0.25)
ridgeModel.fit(X_train, y_train)

# Lasso model, alpha & tol is a hyperparameter
lassoModel = Lasso(alpha = 0.25, tol = 0.0925)
lassoModel.fit(X_train, y_train)
```