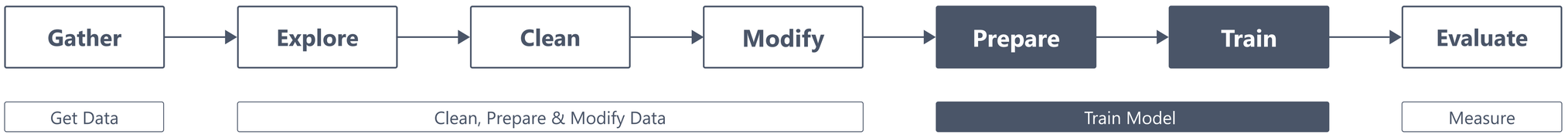
AI Workflow

Model Preparation & Training



PREPARE

DATA PARTITIONING - TRAIN / VALIDATION / HOLDOUT SETS

Before we can create a Machine Learning Model, we need to create our train / validation and holdout sets by partitioning our data. Splitting this data can be done through sklearn or pandas. In the example on the right, you can see how to do this with Pandas.

DATA		
Train (~70% small, ~95% large)	**Validate** (~15% small, ~2.5% large)	**Test** *(= holdout)* (~15% small, ~2.5% large)

Example

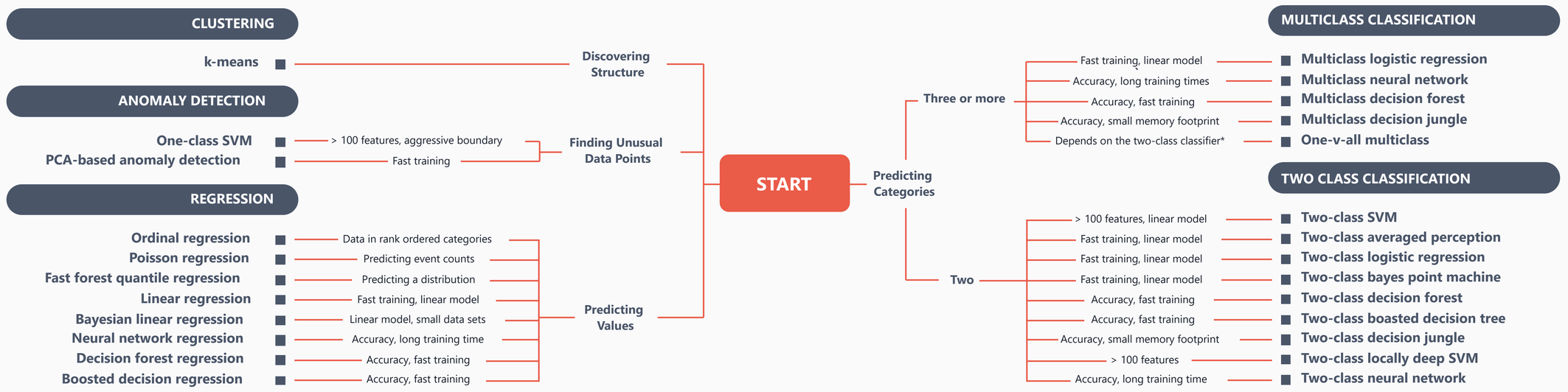
```
> df = pd.DataFrame([['Male', 1], ['Female', 3], ['Female', 2]])
> probs = np.random.rand(len(df))
> msk_training = probs < 0.70
> msk_test = (probs >= 0.7) & (probs < 0.85)
> msk_validation = probs >= 0.85

df_training = df[msk_training]
df_test = df[msk_test]
df_validation = df[msk_validation]
```

TRAINING

ML ALGORITHMS

Below an overview is given for the different Algorithms that are commonly used in Machine Learning. This is far from a complete list, but gives you an idea how how to tackle finding the correct algorithm for the type of task.

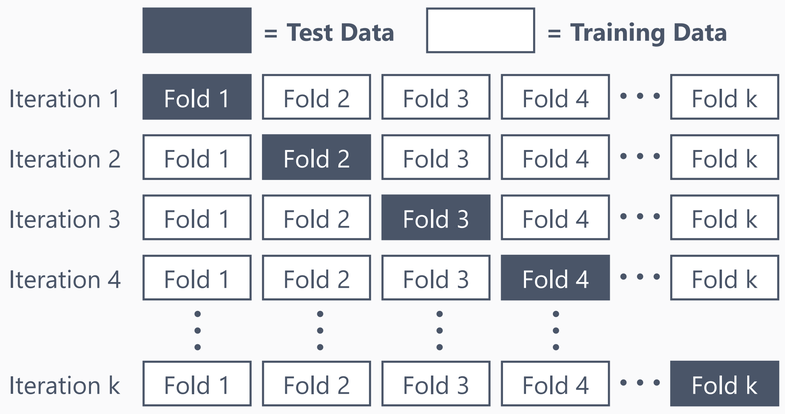


AVOIDING OVERFITTING

Overfitting is the concept of when we trained a model which performs excellent on our training / test set, but performs poorly on sets it hasn't seen before. This for example can happen when tuning our algorithm's hyperparameters until it performs he best, since we are validating it against our test set all the time. Thus tuning the Hyperparameters with leaking knowledge from the test set. One often used way to prevent overfitting is **"Cross Validation"**.

K-FOLD CROSS VALIDATION

In k-fold cross validation, we split the training set in k sets. We then execute the following:
1. Train a model using k - 1 folds
2. Validate the resulting model on the remaining part of the data



AUTOML & GRIDSEARCH

AutoML is the vision of executing the full End-to-End pipeline of data preparation, algorithm selection and algorithm tuning automatically. For algorithm tuning, a technique called "**Grid Search**" is often used to automatically tune the different model hyperparameters.

Example *https://scikit-learn.org/stable/modules/grid_search.html*

```
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV

# Hyperarameter Grid
param_grid = {'C': np.logspace(-5, 8, 15)}

# Initialize Logistic Regression Classifier
model_logreg = LogisticRegression()

# Initialize GridSearch with Cross Validation = 5
model_logreg_cv = GridSearchCV(model_logreg, param_grid, cv=5)

# Fit to data
model_logreg_cv.fit(X, y)

# Tuned Parameters
print("Tuned Logistic Regression Parameters: {}".format(model_logreg_cv.best_params_))
print("Best score is {}".format(model_logreg_cv.best_score_))
```