1 Classification Methods

Classification is the assignment of objects (data points) to categories (classes). It requires a data set (i.e. training set) of points with known class labels. If the class labels are not known you can instead group the data using clustering algorithms (chapter **??**).

1.1 Evaluation of Classifiers

1.1.1 Basic Quality Measures

Accuracy / Success Rate

Precision

Recall

Specificity

Sensitivity

Support

F-score

1.1.2 Area under the Precision-Recall Curve

1.1.3 Handling Unbalanced Data

Having many more samples in one class than the others during training can lead to high accuracy values event hough the classifier performs poorly on the small classes. You can handle the unbalance by:

- up-sampling the smaller data set (creating more artificial samples for that class)
- giving more weight to the samples in the smaller data set

Implementations for Handling Unbalanced Data Sets

Oversampling using imbalanced-learn (see: documentation)

```
from imblearn.over_sampling import RandomOverSampler
ros = RandomOverSampler(random_state=0)
features_resampled, labels_resampled = ros.fit_resample(df[feature_cols],
    df[label_col])
```

Sensitivity, specificity, precision, recall, support and F-score

```
y_true = df[label_col]
y_pred = classifier.predict(df[feature_cols])

from imblearn.metrics import sensitivity_specificity_support
sensitivity, specificity, support = sensitivity_specificity_support(y_true,
    y_pred)

from sklearn.metrics import precision_recall_fscore_support
precision, recall, fscore, support =
    precision_recall_fscore_support(y_true, y_pred)
```

1.2 Linear Classifiers

Linear classifiers use linear decision boundaries to classify points to a respective class.

1.2.1 Support Vector Classifier (SVC)

SVCs use hyperplanes to separate data points according to their class label with a maximum margin (M) between the separating hyperplane ($x^T\beta + \beta_0 = 0$) and the points. If points cannot be perfectly separated by the decision boundary, a soft margin SVM is used with a slack variable ξ that punishes points in the margin or on the wrong side of the hyperplane. The optimization problem is given by [**?**] :

$$\begin{aligned} &\max_{\beta,\beta_0,\beta=1} M, \\ \text{subject to } &y_i(x_i^T\beta + \beta_0) \geq 1 - \xi_i, \quad \forall i, \\ &\xi_i \geq 0, \quad \sum \xi_i \leq constant, \quad i = 1, ..., N, \end{aligned} \tag{1}$$

where β are the coefficients and x are the N data points. The support vectors are the points that determine the orientation of the hyperplane (i.e. the closest points). The classification function is given by:

$$G(x) = \text{sign}[x^T\beta + \beta_0] \tag{2}$$

SVMs are sensitive to the scaling of the features. Therefore, the data should be normalized before classification.

Implementation of SVCs

```
from sklearn import svm
# train the model
svc_model = svm.SVC()
svc_model.fit(train_df[feautre_cols], train_df[label_col])
# test the model
y_predict = svc_model.predict(test_df[feature_cols])
```