Applied Data Science

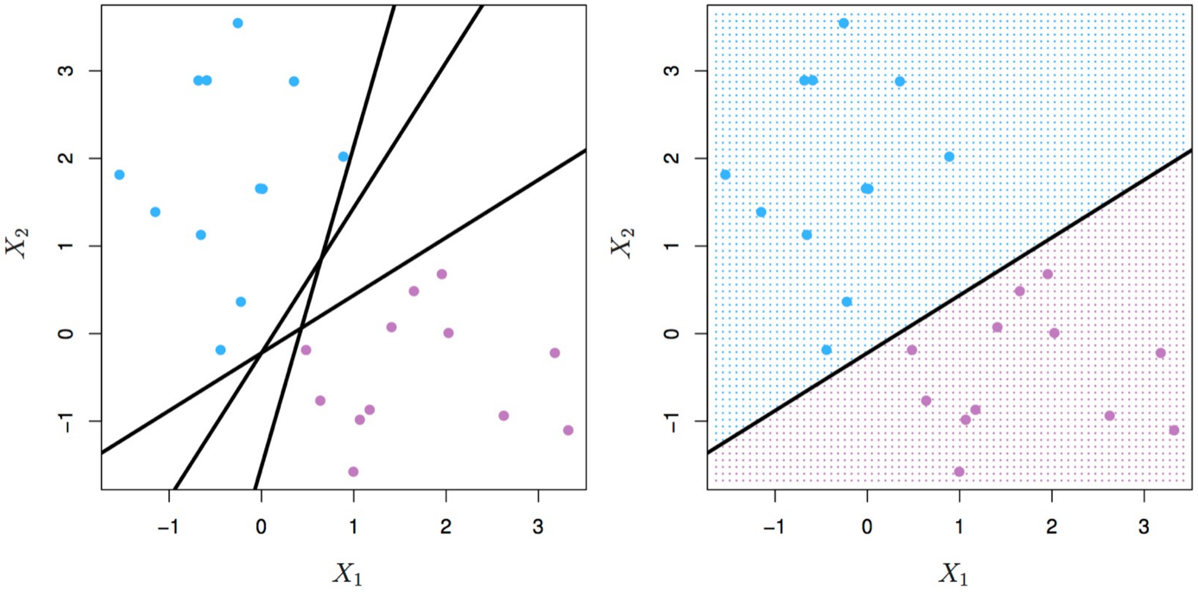
L10. Support vector Machines

Irina Mohorianu

Head of Bioinformatics/ Scientific Computing @CSCI

Support Vector Machines. Maximum margin classifier

Support vector machines (SVMs) are models of supervised learning, applicable to both classification and regression problems. The SVM is an extension of the support vector classifier (SVC), which is turn is an extension of the maximum margin classifier.

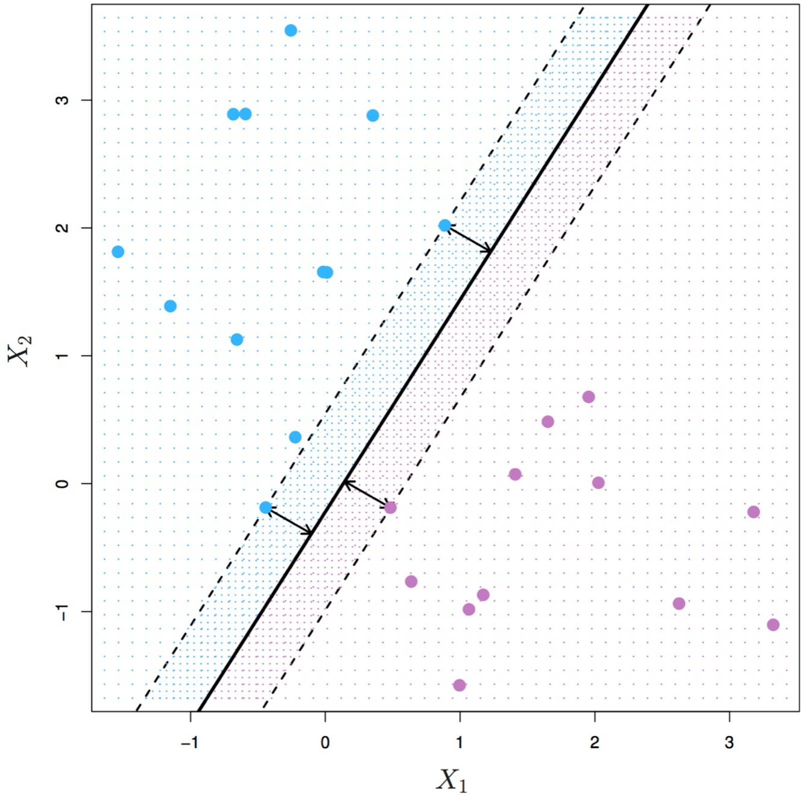


Left: two classes of observations (blue, purple) and three separating hyperplanes. Right: separating hyperplane shown as black line and grid indicates decision rule. Source: http://www-bcf.usc.edu/~gareth/ISL/

Let's start by defining a hyperplane. In *p*-dimensional space, a hyperplane is an affine subspace of *p*-1. The figure below shows three separating hyperplanes and objects of two different classes. A separating hyperplane forms a natural linear decision boundary, classifying new objects according to which side of the line they are located.

Support Vector Machines. Maximum margin classifier

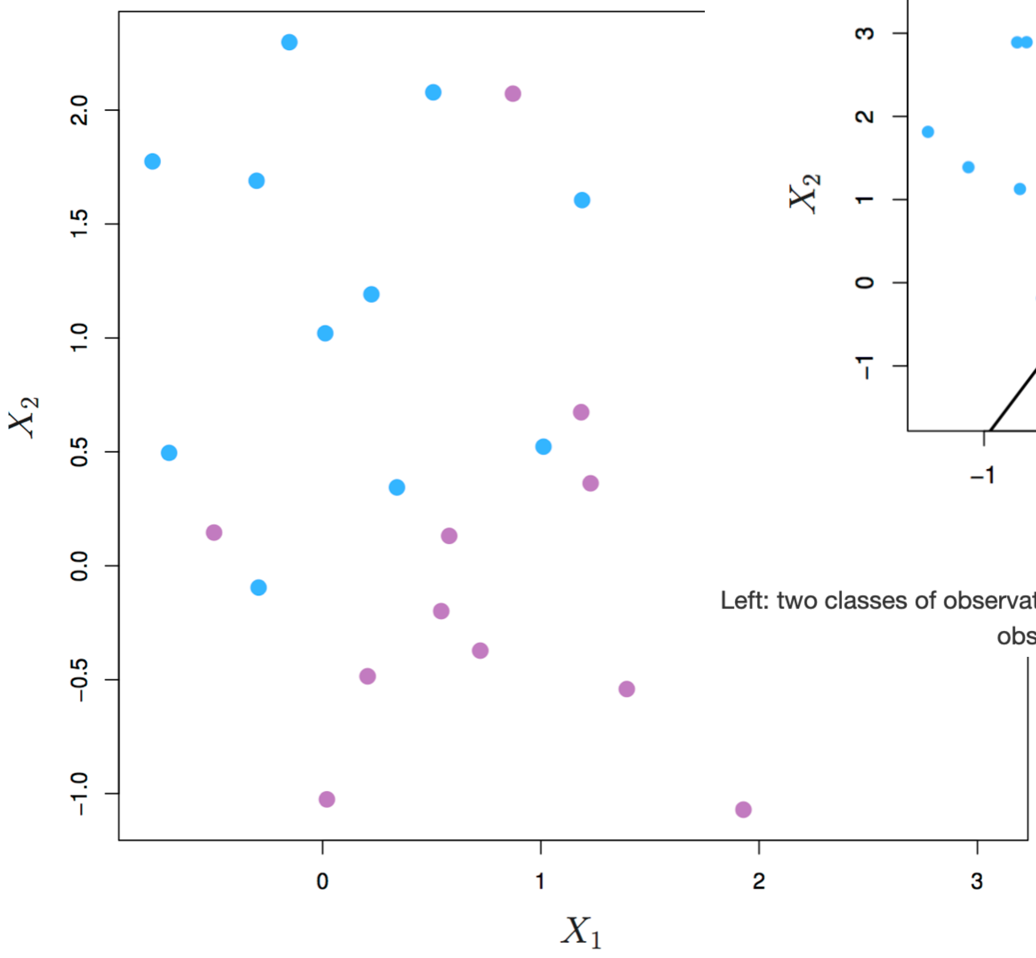
The **maximal margin hyperplane** is the separating hyperplane that is farthest from the training observations. The perpendicular distance from a given hyperplane to the nearest training observation is known as the **margin**. The maximal margin hyperplane is the separating hyperplane for which the margin is largest.

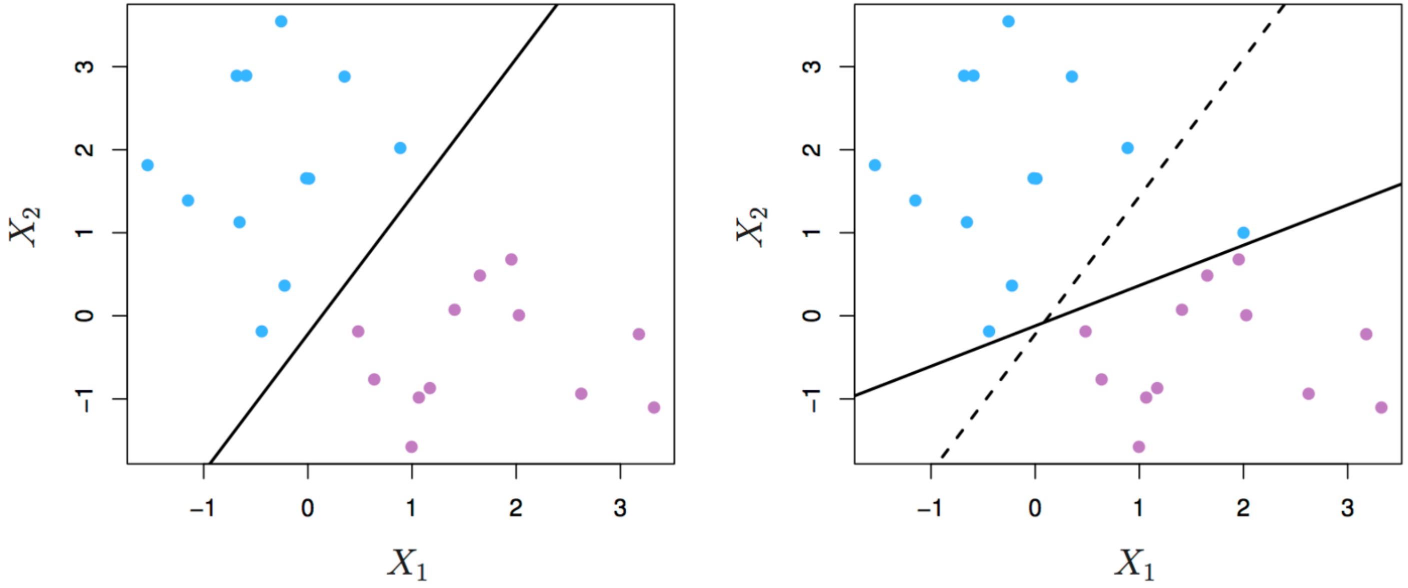


Maximal margin hyperplane shown as solid line. Margin is the distance from the solid line to either of the dashed lines. The support vectors are the points on the dashed line. Source: http://www-bcf.usc.edu/~gareth/ISL/

The figure above shows three training observations that are equidistant from the maximal margin hyperplane and lie on the dashed lines indicating the margin. These are the **support vectors**. If these points were moved slightly, the maximal margin hyperplane would also move, hence the term *support*. The maximal margin hyperplane is set by the **support vectors** alone; it is not influenced by any other observations.

Support Vector Machines.
Support vector classifier



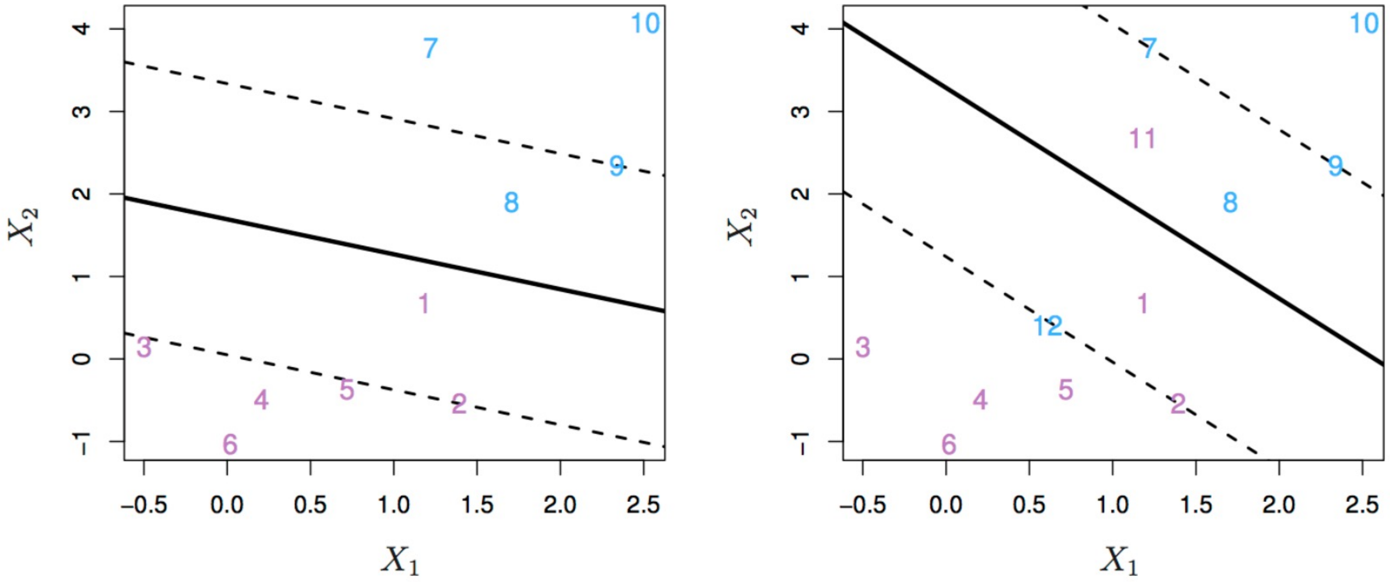


Left: two classes of observations and a maximum margin hyperplane (solid line). Right: Hyperplane (solid line) moves after the addition of a new observation (original hyperplane is dashed line). Source: http://www-bcf.usc.edu/~gareth/ISL/

It would be better to choose a classifier based on a hyperplane that:
- is more robust to individual observations
- provides better classification of most of the training variables

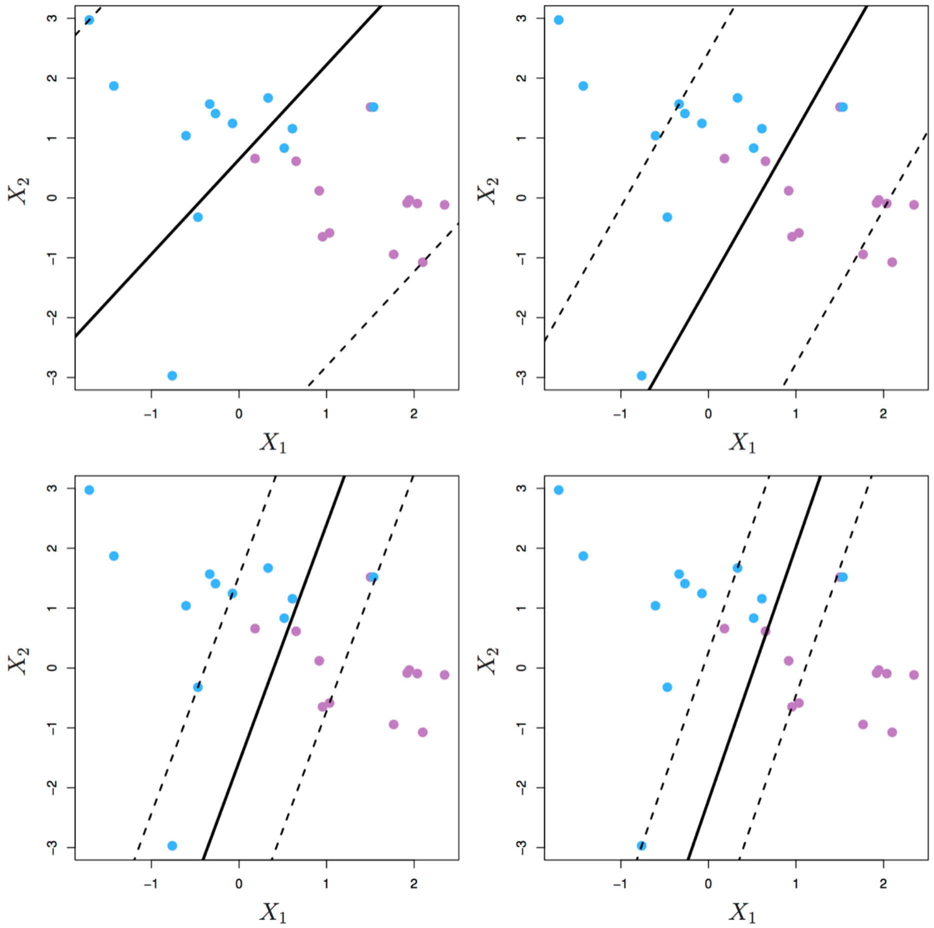
Support Vector Machines.
Support vector classifier

In other words, we might tolerate some misclassifications if the prediction of the remaining observations is more reliable. The **support vector classifier** does this by allowing some observations to be on the wrong side of the margin or even on the wrong side of the hyperplane. Observations on the wrong side of the hyperplane are misclassifications.



Left: observations on the wrong side of the margin. Right: observations on the wrong side of the margin and observations on the wrong side of the hyperplane. Source: http://www-bcf.usc.edu/~gareth/ISL/

Support Vector Machines. Maximum margin classifier



Margin of a support vector classifier changing with tuning parameter C. Largest value of C was used in the top left panel, and smaller values in the top right, bottom left and bottom right panels. Source: http://www-bcf.usc.edu/~gareth/ISL/

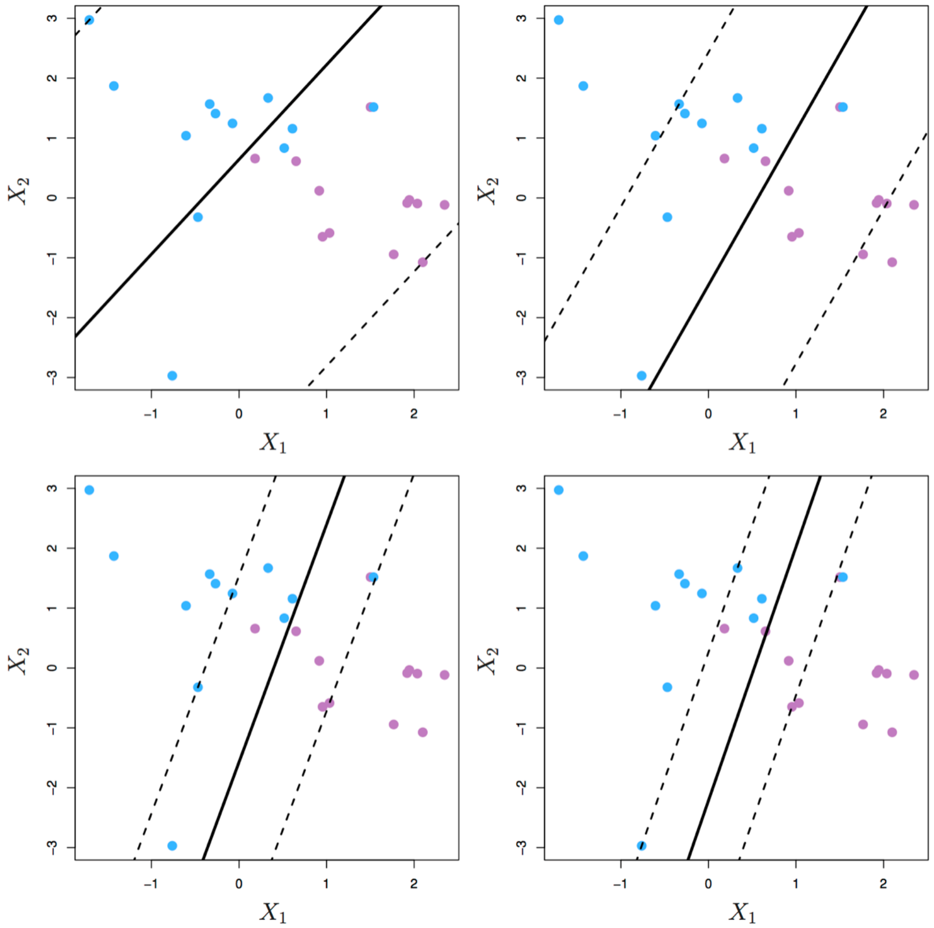
The support vector classifier has a tuning parameter, C, that determines the number/ severity of the margin violations

If $C = 0$, then no violations to the margin will be tolerated, which is equivalent to the maximal margin classifier.

As C increases, the classifier becomes more tolerant of violations to the margin, and so the margin widens.
The optimal value of C is chosen through cross-validation.

C is described as a tuning parameter, because it controls the bias-variance trade-off:
[a] a small C results in narrow margins that are rarely violated; the model will have low bias, but high variance.
[b] as C increases the margins widen allowing more violations; the bias of the model will increase, but its variance will decrease.

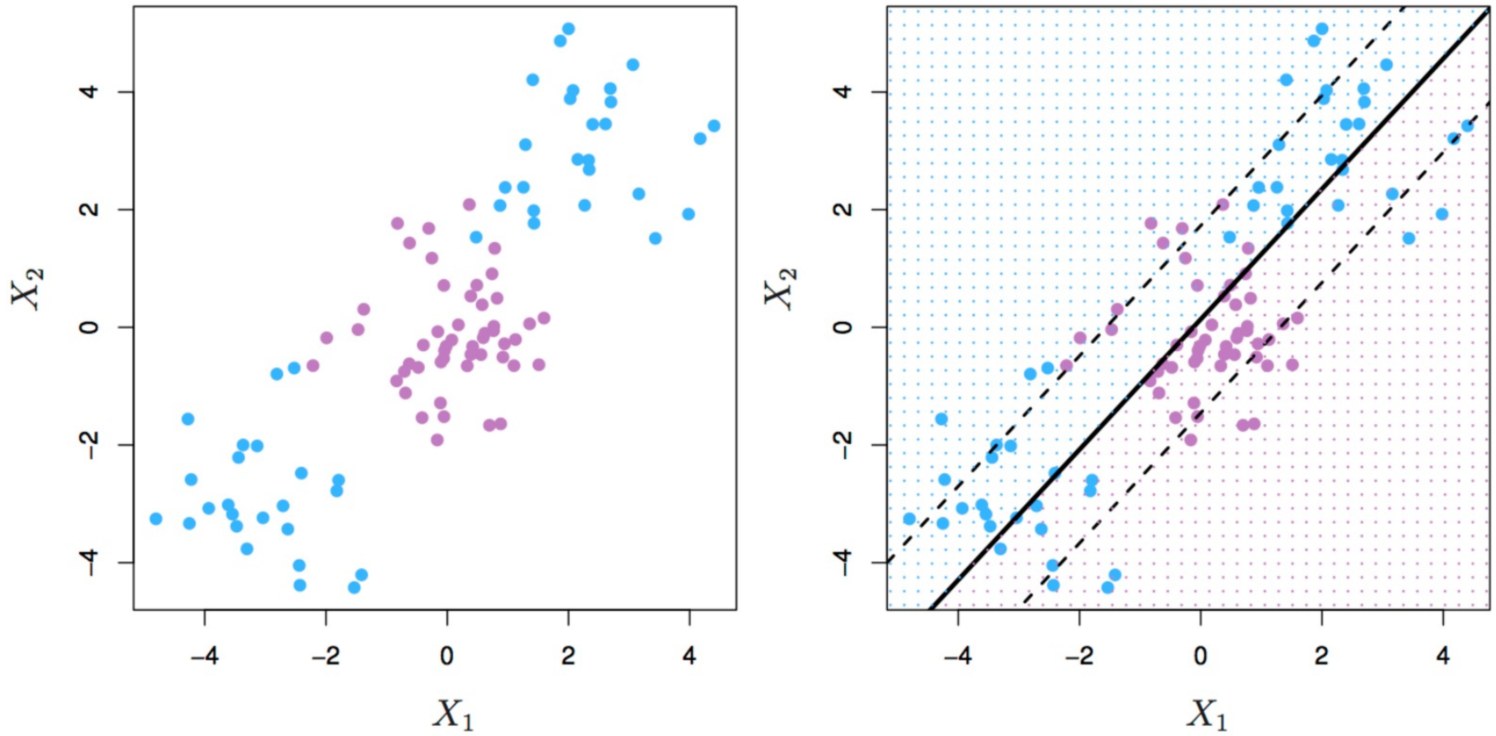
Support Vector Machines. Maximum margin classifier



Margin of a support vector classifier changing with tuning parameter C. Largest value of C was used in the top left panel, and smaller values in the top right, bottom left and bottom right panels. Source: http://www-bcf.usc.edu/~gareth/ISL/

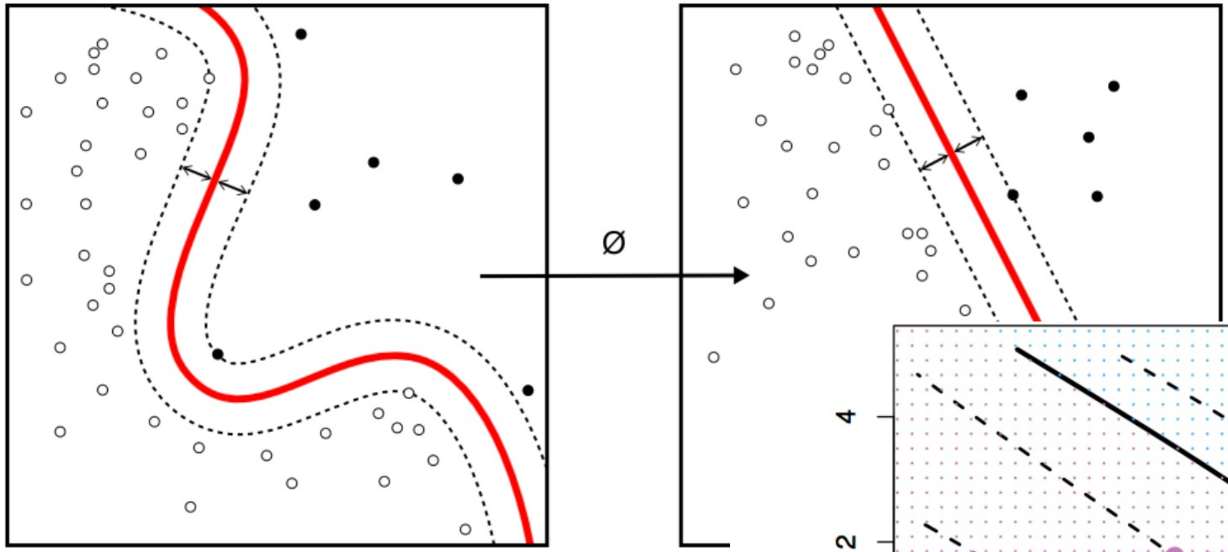
The **support vectors** are the observations that lie directly on the margin, or on the wrong side of the margin for their class. The only observations that affect the classifier are the support vectors. As *C* increases, the margin widens and the number of support vectors increases. In other words, when *C* increases more observations are involved in determining the decision boundary of the classifier.

Support Vector Machines. Non linearly separable classes.

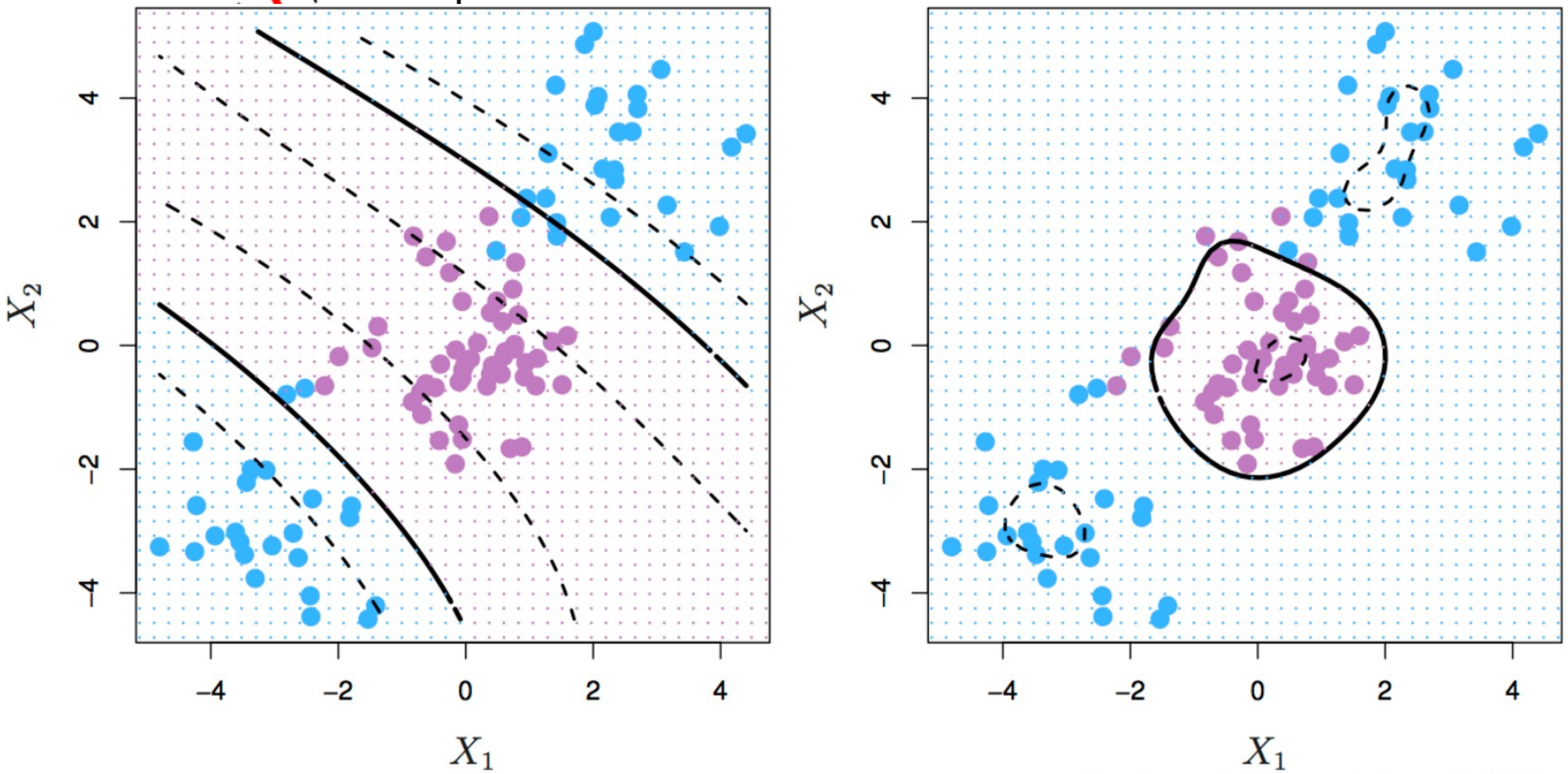


Two classes of observations with a non-linear boundary between them.

Support Vector Machines.



Kernel machine. By Alisneaky - Own work, CC0, https://commons.



Left: SVM with polynomial kernel of degree 3. Right: SVM with radial kernel. Source: http://www-bcf.usc.edu/~gareth/ISL/

Support Vector Machines. Moons example.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, roc_curve, auc
from sklearn.model_selection import RepeatedStratifiedKFold
```

```
[ ] moons = pd.read_csv("moons.csv", header=None)
    moons.columns = ['V1', 'V2', 'V3']

    # Partition data into training and test set
    X = moons[['V1', 'V2']]
    y = moons['V3']
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

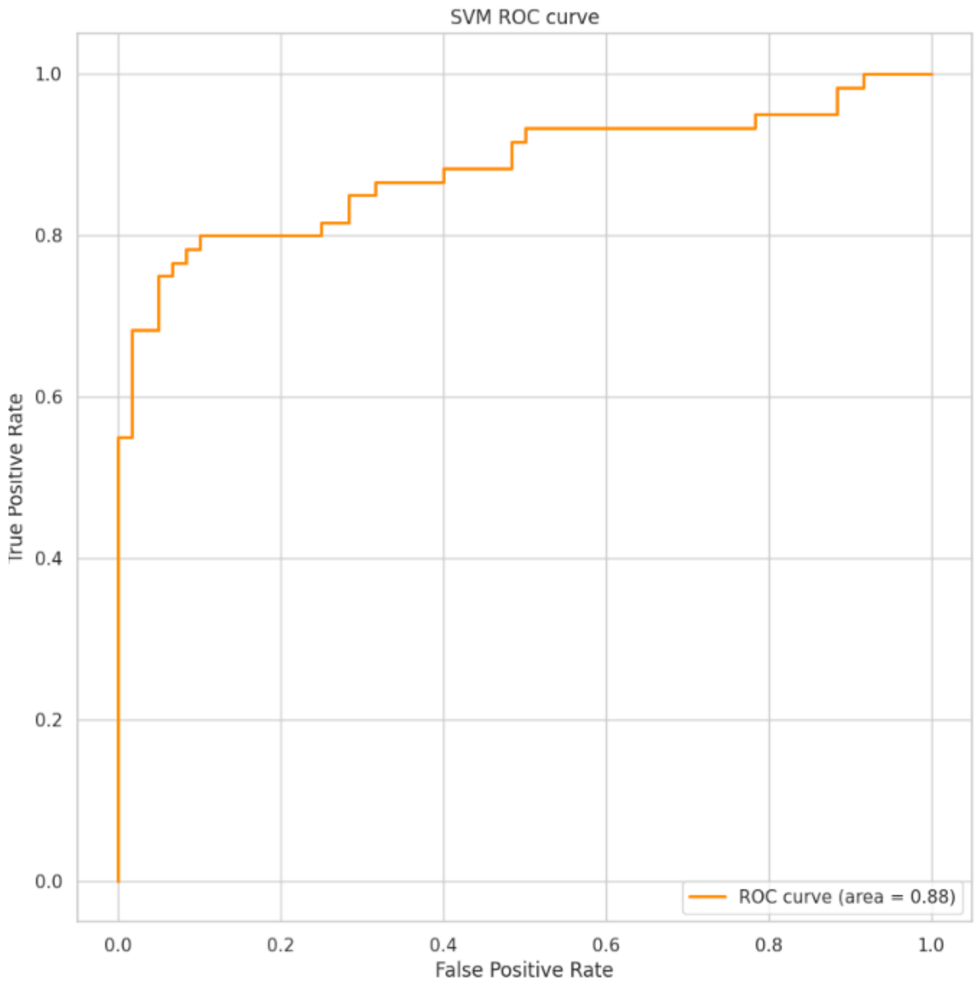
```
[ ] # Visualise data
    sns.set(style="whitegrid")
    sns.scatterplot(data=X_train, x='V1', y='V2', hue=y_train)
    plt.title("Scatterplot of the Training Data")
    plt.show()
```

Support Vector Machines. Moons example.


```
# Set up SVM with Grid Search and Cross-Validation
param_grid = {'C': np.logspace(-2, 2, 9), 'gamma': np.logspace(-2, 2, 9)}

cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=10, random_state=42)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)

svm = SVC(probability=True)
grid_search = GridSearchCV(svm, param_grid, cv=cv, verbose=2, n_jobs=-1)
grid_search.fit(X_train_scaled, y_train)

# Best model
best_svm = grid_search.best_estimator_
```

```
Fitting 100 folds for each of 81 candidates, totalling 8100 fits
```

Support Vector Machines. Moons example.




```
[[55  5]
 [ 9 51]]
              precision    recall  f1-score   support

           A       0.86      0.92      0.89        60
           B       0.91      0.85      0.88        60

    accuracy                           0.88       120
   macro avg       0.89      0.88      0.88       120
weighted avg       0.89      0.88      0.88       120
```

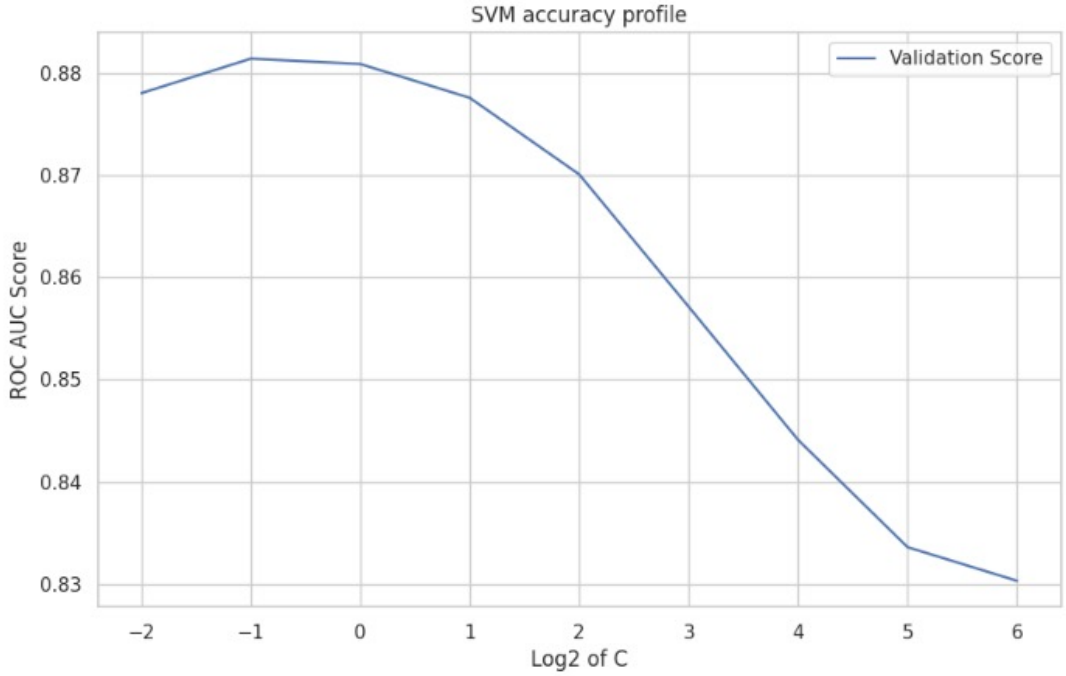
Support Vector Machines.
Cell Segmentation example

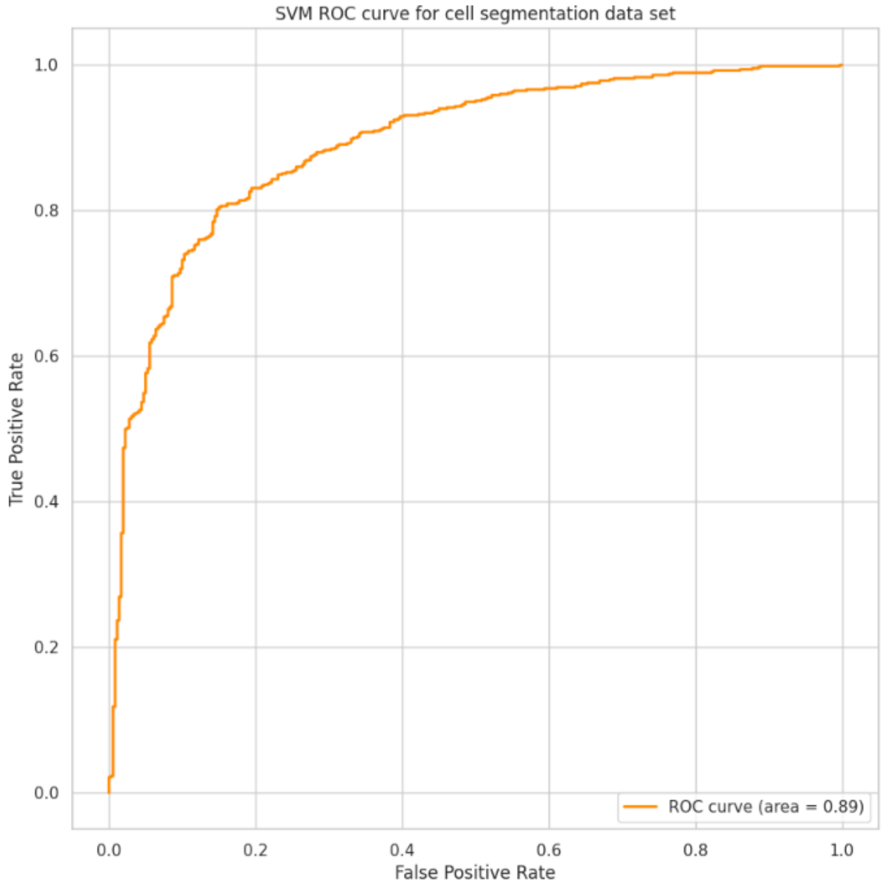
```
[ ] data = pd.read_csv('segmentation_data.csv')
    data = data.drop("Unnamed: 0", axis=1)

    X = data.drop(columns=['Class'])
    X = X.iloc[:,2:]
    y = data['Class']

    # Partition data
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.5, random_state=42, stratify=y)

    # Data preprocessing
    scaler = StandardScaler()
    X_train = scaler.fit_transform(X_train)
    X_test = scaler.transform(X_test)
```

```
# Define grid search parameters and rbf SVM
param_grid = {'C': [0.25, 0.5, 1, 2, 4, 8, 16, 32, 64]}
svm = SVC(kernel="rbf", probability=True)

# Stratified 10-fold 10-repeats CV.
cv = RepeatedStratifiedKFold(n_splits=10, n_repeats=10, random_state=42)
grid_search = GridSearchCV(svm, param_grid, cv=cv, scoring='roc_auc', return_train_score=True, n_jobs=-1)

# Train the SVM model
grid_search.fit(X_train, y_train)
best_svm = grid_search.best_estimator_

print(best_svm)

# Plot SVM accuracy profile
results = pd.DataFrame(grid_search.cv_results_)
plt.figure(figsize=(10, 6))
plt.plot(np.log2(results["param_C"].astype('float32')),
         results["mean_test_score"], label="Validation Score")
plt.xlabel("Log2 of C")
plt.ylabel("ROC AUC Score")
plt.title("SVM accuracy profile")
plt.legend()
plt.show()
```

Support Vector Machines. Cell Segmentation example

```
SVC(C=0.5, probability=True)
```






```
0.891784188034188
```

Acc on the test data: 0.89

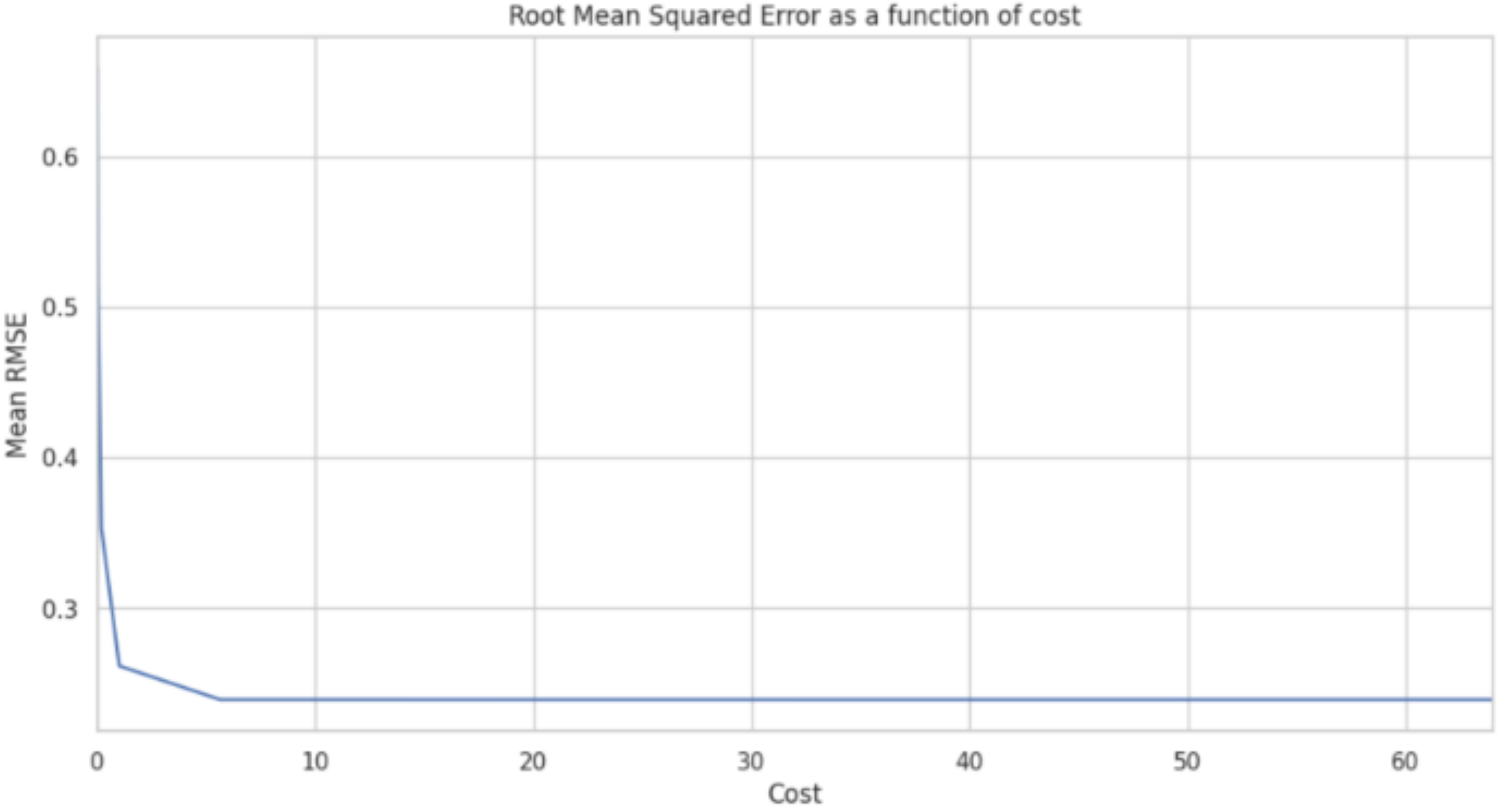
Support Vector Machines. Blood brain barrier example

```
[ ]
from sklearn.svm import SVR
from sklearn.preprocessing import PowerTransformer
from sklearn.feature_selection import VarianceThreshold
from sklearn.pipeline import Pipeline
from sklearn.metrics import make_scorer, mean_squared_error
from sklearn.model_selection import RepeatedKFold

data = pd.read_csv('bbb_df.csv')
descr = data.drop(columns='logBBB')
logBBB = data['logBBB']

train_index, test_index = train_test_split(descr.index, test_size=0.2, random_state=42,)
descr_train = descr.iloc[train_index]
conc_ratio_train = logBBB.iloc[train_index]
descr_test = descr.iloc[-train_index]
conc_ratio_test = logBBB.iloc[-train_index]
```

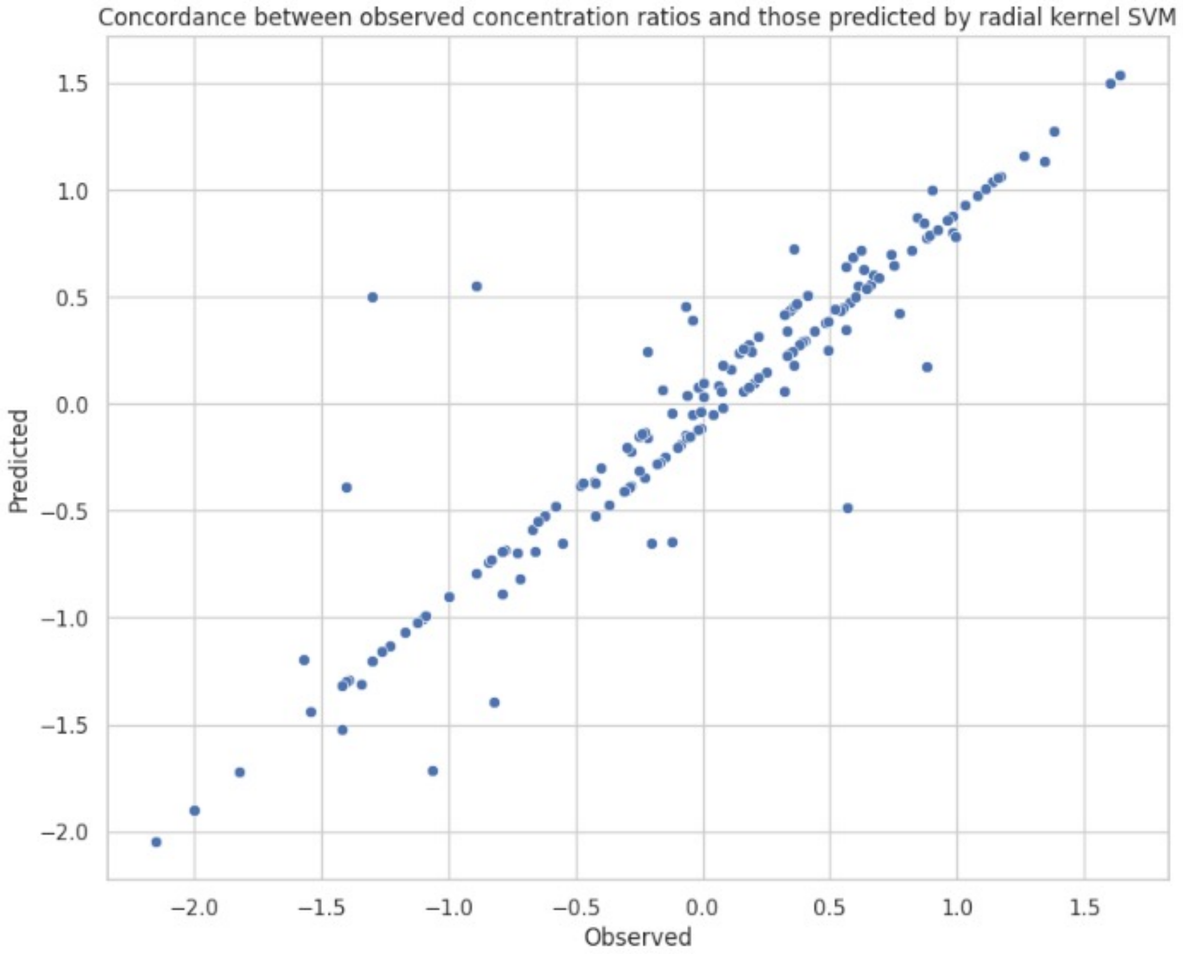
```
# Preprocessing: centering, scaling, correlation filtering, and removing near-zero variance
scaler = StandardScaler()
power_transformer = PowerTransformer(method='yeo-johnson')
variance_threshold = VarianceThreshold(threshold=0.75 * (1 - 0.75))

# Applying transformations to training data
descr_train = pd.DataFrame(scaler.fit_transform(descr_train), columns=descr_train.columns)
descr_train = pd.DataFrame(power_transformer.fit_transform(descr_train), columns=descr_train.columns)
descr_train = pd.DataFrame(variance_threshold.fit_transform(descr_train))

# Fit SVM with radial kernel - 10 CV, 10 repeats.
param_grid = {'C': np.logspace(-3, 3, 9)}
svm = SVR(kernel='rbf')

grid_search = GridSearchCV(svm, param_grid, cv=RepeatedKFold(n_splits=10, n_repeats=10, random_state=42), scoring=make_scorer(mean_squared_error, greater_is_better=False)
grid_search.fit(descr_train, conc_ratio_train)

print(grid_search.best_params_)
```

Optimal value for the C parameter: C = 31.62

Support Vector Machines. Blood brain barrier example



Support Vector Machines. Blood brain barrier example



Correlation between observed and predicted: 0.937