SUPPORT VECTOR MACHINE REGRESSION AND CLASSIFICATION

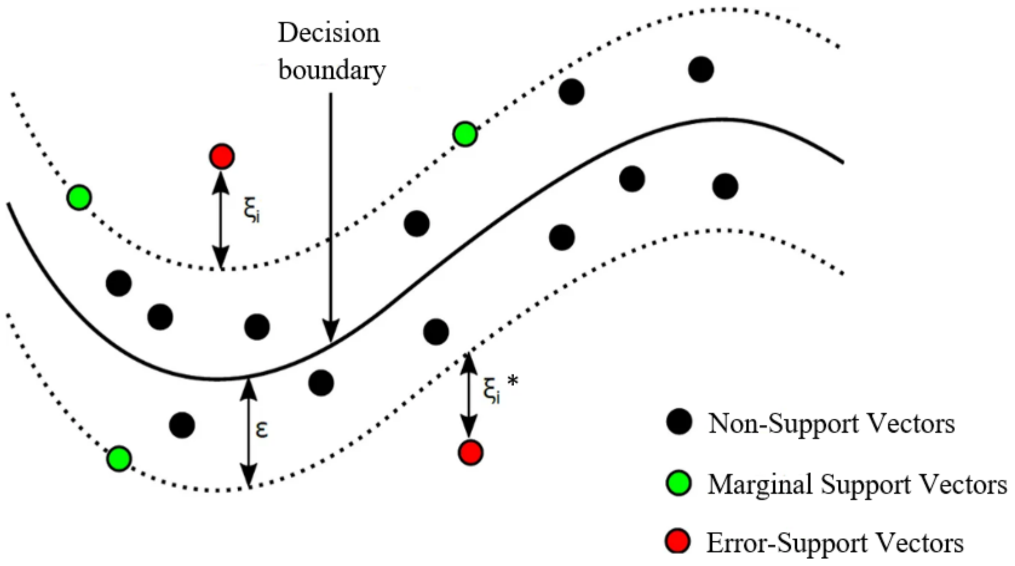
Historical Note. Support vector machine (SVM) analysis is a machine learning tool for regression and classification. It was first proposed by Vladimir Vapnik in his book "The Nature of Statistical Learning Theory", Springer-Verlag, New York, 1995.

Support Vector Regression

The goal of Support Vector Regression is to find a function $f(x_1, ..., x_k)$ that deviates from the observed response y by a value not greater than a pre-specified ε for each training point, and at the same time is as flat as possible.

Let $x = \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1k} \\ & & \cdots & \\ x_{n1} & x_{n2} & \dots & x_{nk} \end{bmatrix}$ be the $n \times k$ matrix of predictor values in the training set. Let $\beta = (\beta_1, \dots, \beta_k)'$ be the column-vector of slopes, and $b = (b_1, \dots, b_n)'$ be the column-vector of intercepts. To find a linear function $f(x) = x\beta + b$ and ensure that it is as flat as possible, we need to find $f(x)$ with the minimal norm $J(\beta) = \frac{1}{2}\beta'\beta$. We also need to observe the constraint that all residuals do not exceed ε, that is, $\left|y_i - \mathrm{x}_i\beta - b\right| \leq \varepsilon, \;\; i = 1, ..., n$, where $\mathrm{x}_i = (x_{i1}, ..., x_{ik})$.

It is possible that no such function f exists. To deal with the infeasible constraints, two non-negative **slack variables** ξ_i and ξ_i^* are introduced for each data point. The objective now is to minimize (the expression is termed the **primal formula**) $J(\beta) = \frac{1}{2}\beta'\beta + C\sum_{i=1}^{n}(\xi_i + \xi_i^*)$ such that $y_i - \mathrm{x}_i\beta - b \leq \varepsilon + \xi_i$, and $\mathrm{x}_i\beta + b - y_i \leq \varepsilon + \xi_i^*$, for all $i = 1, \dots, n$. Here the constant C is the **box constraint**, a positive numeric value that controls the penalty imposed on observations that lie outside the ε-margin and helps to prevent overfitting (it is also known as **regularization parameter**). This value determines the trade-off between the flatness of f and the amount up to which deviations larger than ε are tolerated.



The optimization problem is computationally simpler if formulated in terms of **Lagrange multipliers** α_i and α_i^* for the ith individual, $i = 1, \dots, n$. This leads to minimization of **Lagrangian** (the expression is termed the **dual formula**):

$$L(\alpha) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)\mathrm{x}_i\mathrm{x}_j' \, + \, \varepsilon\sum_{i=1}^{n}\left(\alpha_i + \alpha_i^*\right) \, + \, \sum_{i=1}^{n}\left(\alpha_i^* - \alpha_i\right)y_i,$$

subject to the constraints:

$$\sum_{i=1}^{n}\left(\alpha_i - \alpha_i^*\right) = 0, \quad \text{and} \quad 0 \le \alpha_i, \ \alpha_i^* \le C.$$

The β parameter is found as

$$\beta = \sum_{i=1}^{n}\left(\alpha_i - \alpha_i^*\right)\mathrm{x}_i'.$$

The function f is calculated according to the formula:

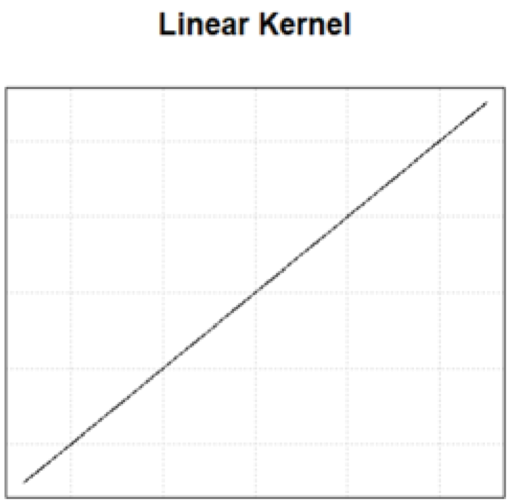
$$f(x) = \sum_{i=1}^{n}\left(\alpha_i - \alpha_i^*\right)\mathrm{x}_i\, x' + b.$$

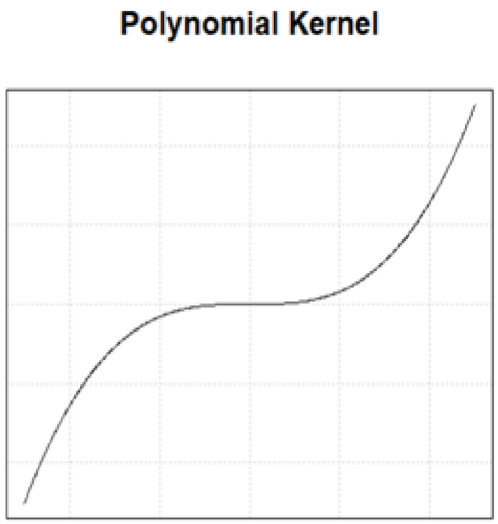
To obtain the optimal solution, the **Karush-Kuhn-Tucker (KKT) complementarity conditions** are used as optimization constraints. For linear support vector regression they are as follows: $\alpha_i(\varepsilon + \xi_i - y_i + \mathrm{x}_i\beta + b) = 0$, $\alpha_i^*(\varepsilon + \xi_i^* + y_i - \mathrm{x}_i\beta - b) = 0$, $\xi_i(C - \alpha_i) = 0$, and $\xi_i^*(C - \alpha_i^*) = 0$, for any $i = 1, \dots, n$. These conditions indicate that all observations strictly inside the epsilon tube have Lagrange multipliers $\alpha_i = \alpha_i^* = 0$. Those observations for which Lagrange multipliers are non-zero (observations on the boundary of the epsilon tube) are called **support vectors**.

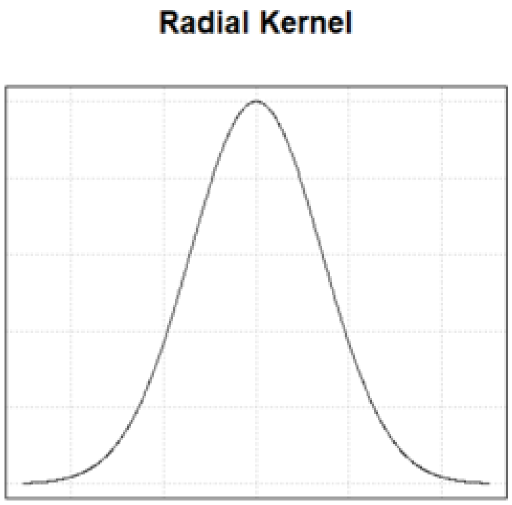
Some regression problems, however, cannot be adequately described using a linear model. In such a case, the Lagrange dual formulation allows it to be extended to nonlinear functions, using kernels. Nonlinear support vector regression finds the coefficients that minimize the Lagrangian

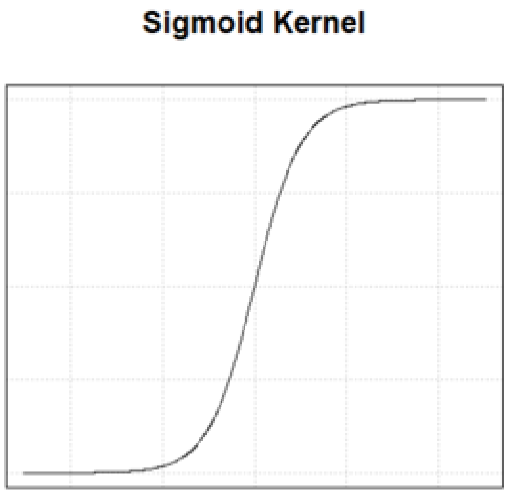
$$L(\alpha) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}(\alpha_i - \alpha_i^*)(\alpha_j - \alpha_j^*)G\big(\mathrm{x}_i,\, \mathrm{x}_j\big) \;+ \varepsilon\sum_{i=1}^{n}\big(\alpha_i + \alpha_i^*\big) \;+ \sum_{i=1}^{n}\big(\alpha_i^* - \alpha_i\big)y_i,$$

where $G\big(\mathrm{x}_i,\, \mathrm{x}_j\big)$ is the kernel function. Several types of kernels are used: $G\big(\mathrm{x}_i,\, \mathrm{x}_j\big) = \mathrm{x}_i\mathrm{x}_j'$ is a **linear kernel**, $G\big(\mathrm{x}_i,\, \mathrm{x}_j\big) = \big(1+\mathrm{x}_i\mathrm{x}_j'\big)^d$ is a **polynomial** kernel of degree d, $G\big(\mathrm{x}_i,\, \mathrm{x}_j\big) = \exp\big(-\gamma\,||\mathrm{x}_i-\mathrm{x}_j||^2\big)$ is a **radial basis function (RBF)** or **radial** or **Gaussian** kernel, and $G\big(\mathrm{x}_i,\, \mathrm{x}_j\big) = tanh(\mathrm{x}_i\mathrm{x}_j')$ is a **sigmoid** kernel, where $tanh(x) = (e^x - e^{-x})/(e^x + e^{-x})$ is the hyperbolic function (see the illustrations below).









Example. We apply the support vector regression to the data in the file "housing_data.csv". The codes below run the analysis and compute the prediction accuracy within 10%, 15%, and 20% of the true values. SAS Enterprise Miner doesn’t handle this method, so we use R and Python only.

In R:

```
housing.data<- read.csv(file="./housing_data.csv", header=TRUE, sep=",")

housing.data$ocean_proximity<- ifelse(housing.data$ocean_proximity==’<1H OCEAN’, 1,
ifelse(housing.data$ocean_proximity==’INLAND’,2,
ifelse(housing.data$ocean_proximity==’NEAR BAY’,3,4)))
```

```
#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
set.seed(234564)
sample <- sample(c(TRUE, FALSE), nrow(housing.data), replace=TRUE, prob=c(0.8,0.2))
train<- housing.data[sample,]
test<- housing.data[!sample,]
test.x<- data.matrix(test[-8])
test.y<- data.matrix(test[8])

install.packages("e1071")
library(e1071)

#FITTING SVR WITH LINEAR KERNEL
svm.reg<- svm(median_house_value ~ housing_median_age+total_rooms+total_bedrooms
+population+households+median_income+ocean_proximity, data=train, kernel="linear")

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
pred.y<- predict(svm.reg, test.x)

#accuracy within 10%
accuracy10<- ifelse(abs(test.y-pred.y)<0.10*test.y,1,0)

#accuracy within 15%
accuracy15<- ifelse(abs(test.y-pred.y)<0.15*test.y,1,0)

#accuracy within 20%
accuracy20<- ifelse(abs(test.y-pred.y)<0.20*test.y,1,0)

print('Linear Kernel')
```

```
"Linear Kernel"
```

```
print(paste('within 10%:', round(mean(accuracy10),4)))
```

```
"within 10%: 0.2212"
```

```
print(paste('within 15%:', round(mean(accuracy15),4)))
```

```
"within 15%: 0.3583"
```

```
print(paste('within 20%:', round(mean(accuracy20),4)))
```

```
"within 20%: 0.5031"
```

```
#FITTING SVR WITH POLYNOMIAL KERNEL
svm.reg<- svm(median_house_value ~ housing_median_age+total_rooms+total_bedrooms
+population+households+median_income+ocean_proximity, + data=train, kernel="poly")

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
pred.y<- predict(svm.reg, test.x)

#accuracy within 10%
accuracy10<- ifelse(abs(test.y-pred.y)<0.10*test.y,1,0)

#accuracy within 15%
accuracy15<- ifelse(abs(test.y-pred.y)<0.15*test.y,1,0)

#accuracy within 20%
accuracy20<- ifelse(abs(test.y-pred.y)<0.20*test.y,1,0)

print('Polynomial Kernel')
```

```
"Polynomial Kernel"
```

```
print(paste('within 10%:', round(mean(accuracy10),4)))
```

```
"within 10%: 0.2414"
```

```
print(paste('within 15%:', round(mean(accuracy15),4)))
```

```
"within 15%: 0.3536"
```

```
print(paste('within 20%:', round(mean(accuracy20),4)))
```

```
"within 20%: 0.4408"
```

```
#FITTING SVR WITH RADIAL KERNEL
svm.reg<- svm(median_house_value ~ housing_median_age+total_rooms+total_bedrooms
+population+households+median_income+ocean_proximity,+ data=train, kernel="radial")

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
pred.y<- predict(svm.reg, test.x)
```

```
#accuracy within 10%
accuracy10<- ifelse(abs(test.y-pred.y)<0.10*test.y,1,0)

#accuracy within 15%
accuracy15<- ifelse(abs(test.y-pred.y)<0.15*test.y,1,0)

#accuracy within 20%
accuracy20<- ifelse(abs(test.y-pred.y)<0.20*test.y,1,0)

print('Radial Kernel')
```

```
"Radial Kernel"
```

```
print(paste('within 10%:', round(mean(accuracy10),4)))
```

```
"within 10%: 0.3676"
```

```
print(paste('within 15%:', round(mean(accuracy15),4)))
```

```
"within 15%: 0.5249"
```

```
print(paste('within 20%:', round(mean(accuracy20),4)))
```

```
"within 20%: 0.6526"
```

```
#FITTING SVR WITH SIGMOID KERNEL
svm.reg<- svm(median_house_value ~ housing_median_age+total_rooms+total_bedrooms
+population+households+median_income+ocean_proximity, + data=train, kernel="sigmoid")

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
pred.y<- predict(svm.reg, test.x)

#accuracy within 10%
accuracy10<- ifelse(abs(test.y-pred.y)<0.10*test.y,1,0)

#accuracy within 15%
accuracy15<- ifelse(abs(test.y-pred.y)<0.15*test.y,1,0)

#accuracy within 20%
accuracy20<- ifelse(abs(test.y-pred.y)<0.20*test.y,1,0)

print('Sigmoid Kernel')
```

```
"Sigmoid Kernel"
```

```
print(paste('within 10%:', round(mean(accuracy10),4)))
```

```
"within 10%: 0.0047"
```

```
print(paste('within 15%:', round(mean(accuracy15),4)))
```

```
"within 15%: 0.0093"
```

```
print(paste('within 20%:', round(mean(accuracy20),4)))
```

```
"within 20%: 0.014"
```

We see that the support vector regression with the radial kernel has the best prediction accuracy, thus, it's the best-fitted model.

In Python:

```python
import pandas
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR
from statistics import mean

housing=pandas.read_csv('C:/Users/000110888/OneDrive - CSULB/Desktop/housing_data.csv')
coding={'<1H OCEAN': 1, 'INLAND': 2, 'NEAR BAY': 3, 'NEAR OCEAN': 4}
housing['ocean_proximity']=housing['ocean_proximity'].map(coding)
X=housing.iloc[:,0:7].values
y=housing.iloc[:,7].values

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.20,
random_state=445021)

#########################################################################################
#FITTING SUPPORT VECTOR REGRESSION WITH LINEAR KERNEL
svreg_linear=SVR(kernel='linear').fit(X_train, y_train)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=svreg_linear.predict(X_test)

ind10=[]
ind15=[]
ind20=[]

for sub1, sub2 in zip(y_pred, y_test):
    ind10.append(1) if abs(sub1-sub2)<0.10*sub2 else ind10.append(0)
    ind15.append(1) if abs(sub1-sub2)<0.15*sub2 else ind15.append(0)
    ind20.append(1) if abs(sub1-sub2)<0.20*sub2 else ind20.append(0)

print('Linear Kernel Prediction Accuracy')
#accuracy within 10%
print('within 10%:', round(mean(ind10),4))

#accuracy within 15%
print('within 15%:', round(mean(ind15),4))
```

```
#accuracy within 20%
print('within 20%:', round(mean(ind20),4))

#########################################################################################
#FITTING SUPPORT VECTOR REGRESSION WITH POLYNOMIAL KERNEL
svreg_poly=SVR(kernel='poly').fit(X_train, y_train)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=svreg_poly.predict(X_test)

ind10=[]
ind15=[]
ind20=[]

for sub1, sub2 in zip(y_pred, y_test):
    ind10.append(1) if abs(sub1-sub2)<0.10*sub2 else ind10.append(0)
    ind15.append(1) if abs(sub1-sub2)<0.15*sub2 else ind15.append(0)
    ind20.append(1) if abs(sub1-sub2)<0.20*sub2 else ind20.append(0)

print('')
print('Polynomial Kernel Prediction Accuracy')
#accuracy within 10%
print('within 10%:', round(mean(ind10),4))

#accuracy within 15%
print('within 15%:', round(mean(ind15),4))

#accuracy within 20%
print('within 20%:', round(mean(ind20),4))


#########################################################################################
#FITTING SUPPORT VECTOR REGRESSION WITH RADIAL KERNEL
svreg_radial=SVR(kernel='rbf').fit(X_train, y_train)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=svreg_radial.predict(X_test)

ind10=[]
ind15=[]
ind20=[]
```

```
for sub1, sub2 in zip(y_pred, y_test):
    ind10.append(1) if abs(sub1-sub2)<0.10*sub2 else ind10.append(0)
    ind15.append(1) if abs(sub1-sub2)<0.15*sub2 else ind15.append(0)
    ind20.append(1) if abs(sub1-sub2)<0.20*sub2 else ind20.append(0)

print('')
print('Radial Kernel Prediction Accuracy')
#accuracy within 10%
print('within 10%:', round(mean(ind10),4))

#accuracy within 15%
print('within 15%:', round(mean(ind15),4))

#accuracy within 20%
print('within 20%:', round(mean(ind20),4))

##########################################################################################
#FITTING SUPPORT VECTOR REGRESSION WITH SIGMOID KERNEL
svreg_sigmoid=SVR(kernel='sigmoid').fit(X_train, y_train)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=svreg_sigmoid.predict(X_test)

ind10=[]
ind15=[]
ind20=[]

for sub1, sub2 in zip(y_pred, y_test):
    ind10.append(1) if abs(sub1-sub2)<0.10*sub2 else ind10.append(0)
    ind15.append(1) if abs(sub1-sub2)<0.15*sub2 else ind15.append(0)
    ind20.append(1) if abs(sub1-sub2)<0.20*sub2 else ind20.append(0)

print('')
print('Sigmoid Kernel Prediction Accuracy')
#accuracy within 10%
print('within 10%:', round(mean(ind10),4))

#accuracy within 15%
print('within 15%:', round(mean(ind15),4))

#accuracy within 20%
print('within 20%:', round(mean(ind20),4))
```

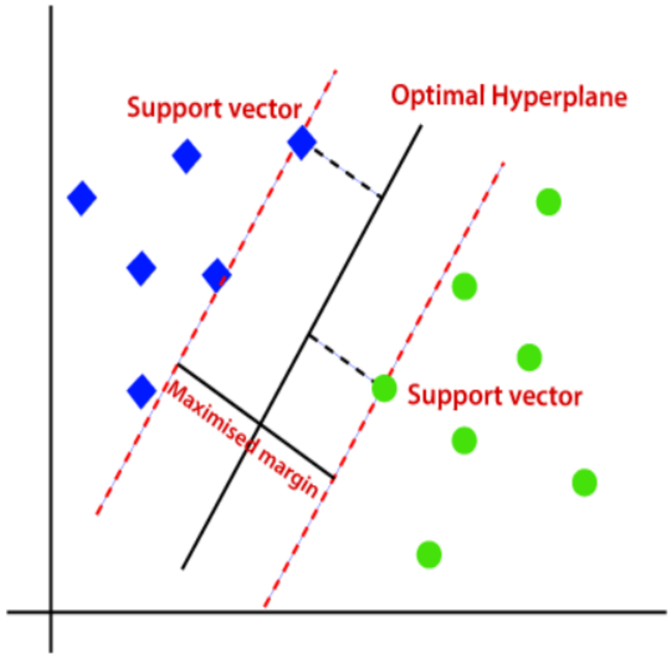
```
Linear Kernel Prediction Accuracy
within 10%: 0.1494
within 15%: 0.2267
within 20%: 0.2953
```

```
Polynomial Kernel Prediction Accuracy
within 10%: 0.1019
within 15%: 0.1634
within 20%: 0.2179

Radial Kernel Prediction Accuracy
within 10%: 0.1002
within 15%: 0.1634
within 20%: 0.2197

Sigmoid Kernel Prediction Accuracy
within 10%: 0.0967
within 15%: 0.1617
within 20%: 0.2179
```

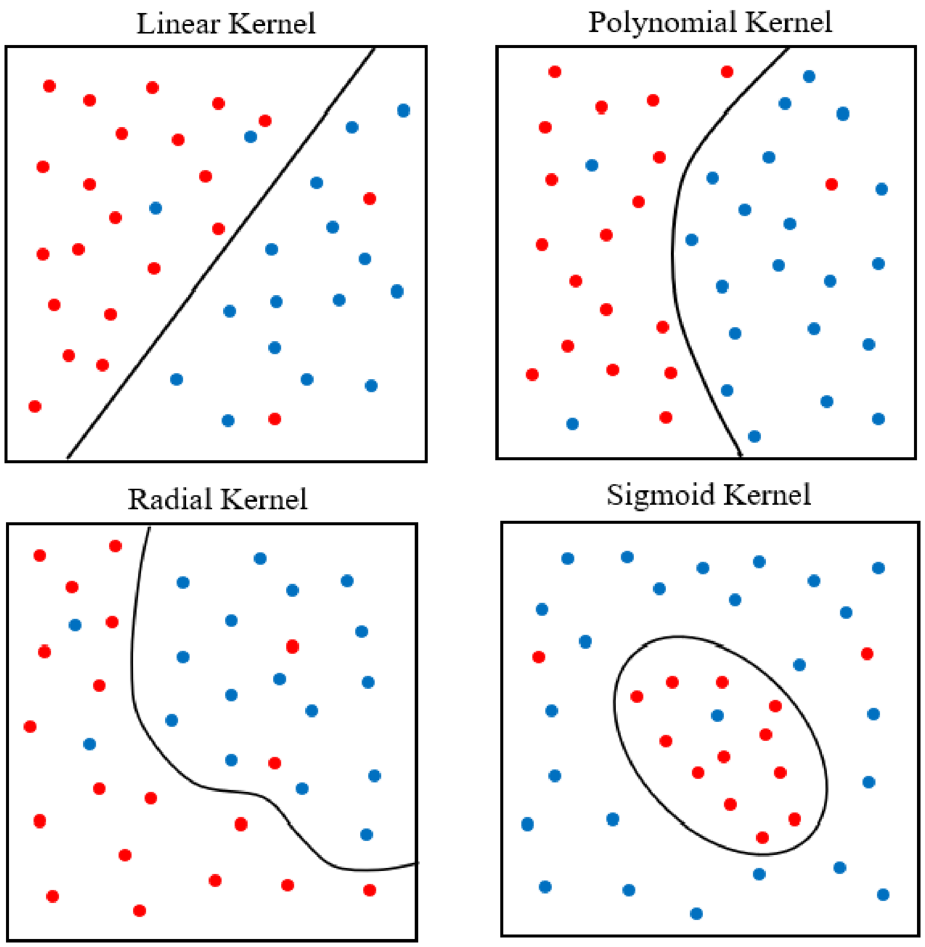
Comparing prediction accuracies, we see that the SVR with the linear kernel fits the data the best. $\square$

Support Vector Machine for Binary Classifier

For binary response, a support vector machine classifies data by finding the best hyperplane that separates data points of one class from those of the other class. The best hyperplane for an SVM is the one with the largest margin between the two classes. Margin means the maximal width of the slab parallel to the hyperplane that has no interior data points. The **support vectors** are the data points that are closest to the separating hyperplane; these points are on the boundary of the slab.

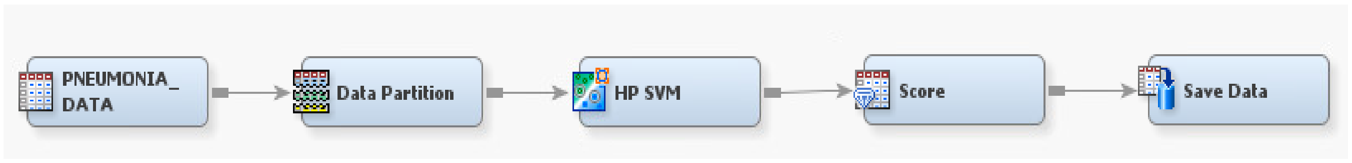


In mathematical terms, the response variable $y_i = \pm 1$ represents the category that the ith individual belongs to. The hyperplane has the equation $f(x) = x\beta + b = 0$. To find the best separating hyperplane, we find β and b that minimize $\beta'\beta$ such that for all $i = 1, \ldots, n$, $y_i f(x_i) \geq 1$. The **support vectors** are the points on the boundary, that is, those for which $y_i f(x_i) = 1$. The dual formulation in this setting is to maximize with respect to α $\sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \, y_i \, y_j x_i x_j'$, subject to the constraints $\sum_{i=1}^{n} y_i \alpha_i = 0$, and $0 \leq \alpha_i \leq C, \quad i = 1, \ldots, n$. Some binary classification problems can't be solved with a simple hyperplane. In this case, kernels (polynomial, radial, or sigmoid) are used. The resulting separating borders are schematically depicted in the figure below. Note different shapes for different kernels.



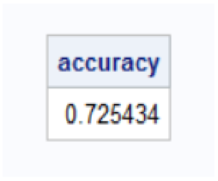
Example. We use the support vector machine binary classifier for the data "pneumonia_data.csv".

In SAS: In Enterprise Miner, we use the following path diagram that includes the HP SVM node. The only choices for kernels are polynomial (degree 2 or higher), radial, and sigmoid, which are specified by changing "Optimization Method" to "Active Set" and choosing a kernel under "Active Set Options".



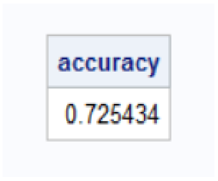
We run the SAS code given below to compute prediction accuracy for the three models.

```
data polynomial_kernel;
set './polynomial_kernel.sas7bdat';
match=(pneumonia=lowcase(EM_CLASSIFICATION));
run;

proc sql;
select mean(match) as accuracy
from polynomial_kernel;
run;
```



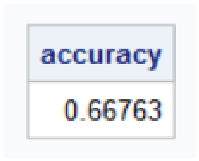
```
accuracy
0.725434
```

```
data radial_kernel;
set './radial_kernel.sas7bdat';
match=(pneumonia=lowcase(EM_CLASSIFICATION));
run;

proc sql;
select mean(match) as accuracy
from radial_kernel;
run;
```



```
accuracy
0.725434
```

```
data sigmoid_kernel;
set './sigmoid_kernel.sas7bdat';
match=(pneumonia=lowcase(EM_CLASSIFICATION));
run;

proc sql;
select mean(match) as accuracy
from sigmoid_kernel;
run;
```



accuracy
0.66763

Models with polynomial (quadratic) and radial kernels have the largest prediction accuracy.

In R:

```
pneumonia.data<- read.csv(file="./pneumonia_data.csv", header=TRUE, sep=",")

pneumonia.data$pneumonia<- ifelse(pneumonia.data$pneumonia=="yes",1,0)
pneumonia.data$gender<- ifelse(pneumonia.data$gender=='M',1,0)
pneumonia.data$tobacco_use<- ifelse(pneumonia.data$tobacco_use=='yes',1,0)

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
set.seed(966452)
sample <- sample(c(TRUE, FALSE), nrow(pneumonia.data), replace=TRUE, prob=c(0.8,0.2))
train<- pneumonia.data[sample,]
test<- pneumonia.data[!sample,]

train.x<- data.matrix(train[-5])
train.y<- data.matrix(train[5])
test.x<- data.matrix(test[-5])
test.y<- data.matrix(test[5])

library(e1071)

#FITTING SVM WITH LINEAR KERNEL
svm.class<- svm(as.factor(pneumonia) ~ gender + age + tobacco_use + PM2_5, data=train, ker-
nel="linear")

#computing prediction accuracy for testing data
pred.y<- as.numeric(predict(svm.class, test.x))-1

for (i in 1:length(pred.y))
   match[i]<- ifelse(test.y[i]==pred.y[i], 1,0)
print(paste("accuracy=", round(mean(match), digits=4)))
```

```
"accuracy= 0.7216"
```

```
#FITTING SVM WITH POLYNOMIAL KERNEL
svm.class<- svm(as.factor(pneumonia) ~ gender + age + tobacco_use + PM2_5, data=train, ker-
nel="polynomial")

#computing prediction accuracy for testing data
pred.y<- as.numeric(predict(svm.class, test.x))-1

for (i in 1:length(pred.y))
    match[i]<- ifelse(test.y[i]==pred.y[i], 1,0)
print(paste("accuracy=", round(mean(match), digits=4)))
```

```
"accuracy= 0.7335"
```

```
#FITTING SVM WITH RADIAL KERNEL
svm.class<- svm(as.factor(pneumonia) ~ gender + age + tobacco_use + PM2_5, data=train, ker-
nel="radial")

#computing prediction accuracy for testing data
pred.y<- as.numeric(predict(svm.class, test.x))-1

for (i in 1:length(pred.y))
    match[i]<- ifelse(test.y[i]==pred.y[i], 1,0)
print(paste("accuracy=", round(mean(match), digits=4)))
```

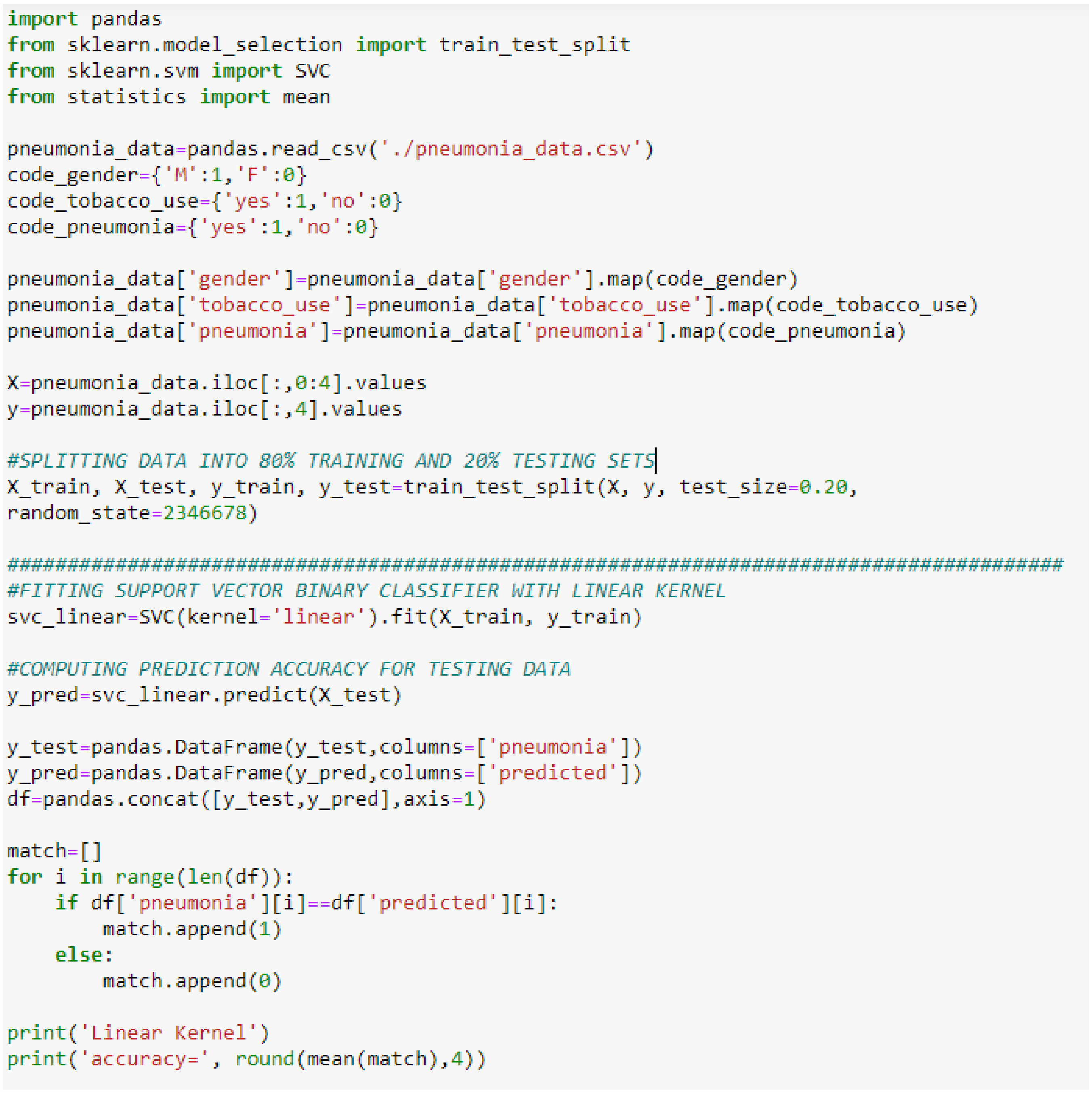
```
"accuracy= 0.7425"
```

```
#FITTING SVM WITH SIGMOID KERNEL
svm.class<- svm(as.factor(pneumonia) ~ gender + age + tobacco_use + PM2_5, data=train, ker-
nel="sigmoid")

#computing prediction accuracy for testing data
pred.y<- as.numeric(predict(svm.class, test.x))-1

for (i in 1:length(pred.y))
    match[i]<- ifelse(test.y[i]==pred.y[i], 1,0)
print(paste("accuracy=", round(mean(match), digits=4)))
```

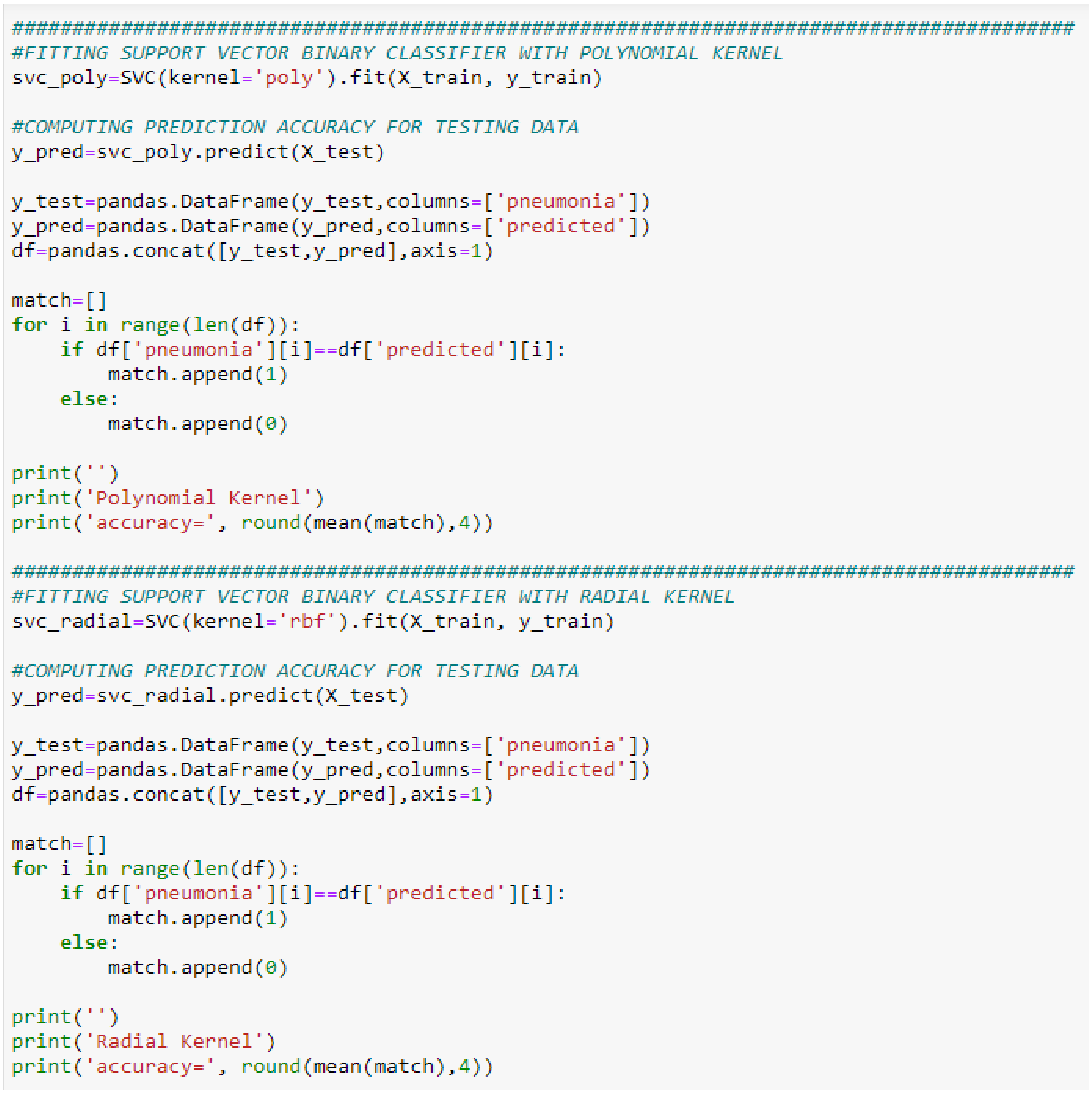
```
"accuracy= 0.6257"
```

The model with the radial kernel has the largest accuracy of prediction.

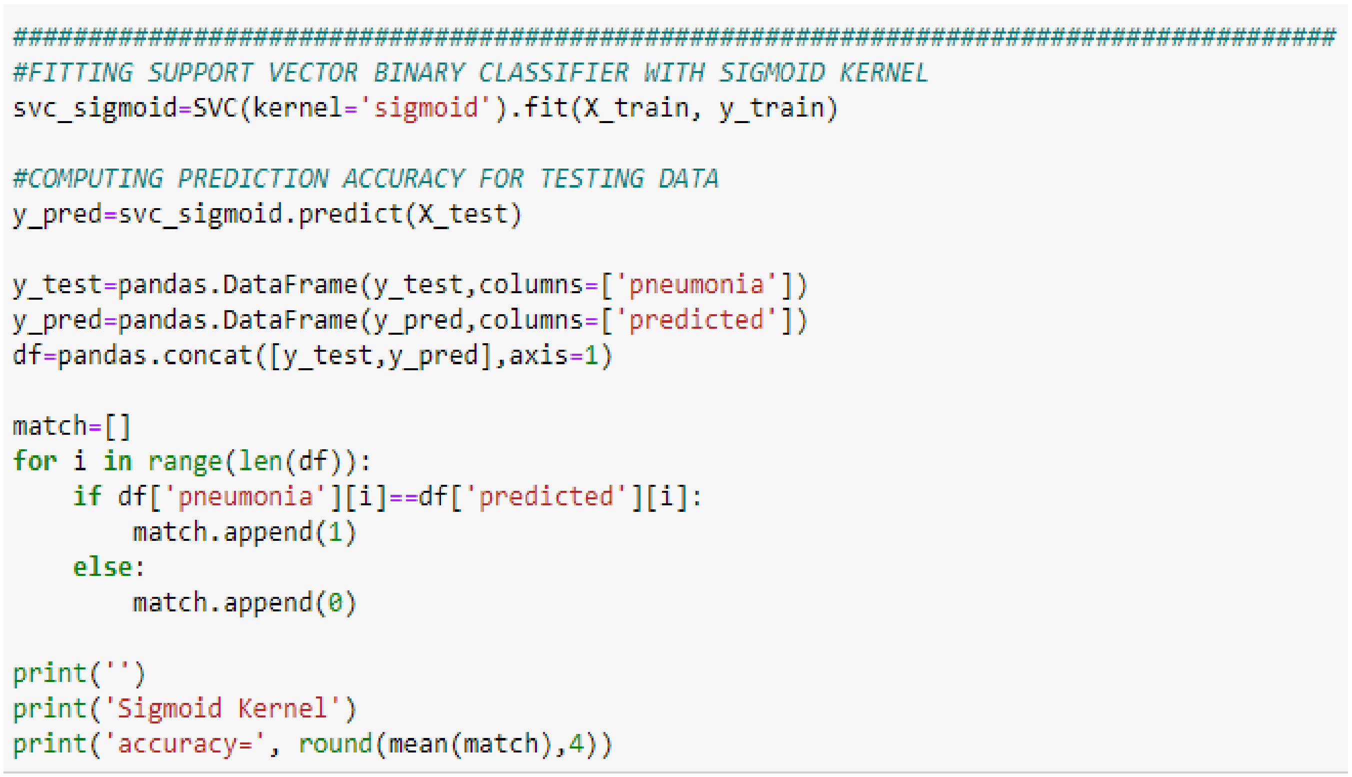
In Python:



```python
import pandas
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from statistics import mean

pneumonia_data=pandas.read_csv('./pneumonia_data.csv')
code_gender={'M':1,'F':0}
code_tobacco_use={'yes':1,'no':0}
code_pneumonia={'yes':1,'no':0}

pneumonia_data['gender']=pneumonia_data['gender'].map(code_gender)
pneumonia_data['tobacco_use']=pneumonia_data['tobacco_use'].map(code_tobacco_use)
pneumonia_data['pneumonia']=pneumonia_data['pneumonia'].map(code_pneumonia)

X=pneumonia_data.iloc[:,0:4].values
y=pneumonia_data.iloc[:,4].values

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.20,
random_state=2346678)

###########################################################################################
#FITTING SUPPORT VECTOR BINARY CLASSIFIER WITH LINEAR KERNEL
svc_linear=SVC(kernel='linear').fit(X_train, y_train)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=svc_linear.predict(X_test)

y_test=pandas.DataFrame(y_test,columns=['pneumonia'])
y_pred=pandas.DataFrame(y_pred,columns=['predicted'])
df=pandas.concat([y_test,y_pred],axis=1)

match=[]
for i in range(len(df)):
    if df['pneumonia'][i]==df['predicted'][i]:
        match.append(1)
    else:
        match.append(0)

print('Linear Kernel')
print('accuracy=', round(mean(match),4))
```



```
##########################################################################################
#FITTING SUPPORT VECTOR BINARY CLASSIFIER WITH POLYNOMIAL KERNEL
svc_poly=SVC(kernel='poly').fit(X_train, y_train)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=svc_poly.predict(X_test)

y_test=pandas.DataFrame(y_test,columns=['pneumonia'])
y_pred=pandas.DataFrame(y_pred,columns=['predicted'])
df=pandas.concat([y_test,y_pred],axis=1)

match=[]
for i in range(len(df)):
    if df['pneumonia'][i]==df['predicted'][i]:
        match.append(1)
    else:
        match.append(0)

print('')
print('Polynomial Kernel')
print('accuracy=', round(mean(match),4))

##########################################################################################
#FITTING SUPPORT VECTOR BINARY CLASSIFIER WITH RADIAL KERNEL
svc_radial=SVC(kernel='rbf').fit(X_train, y_train)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=svc_radial.predict(X_test)

y_test=pandas.DataFrame(y_test,columns=['pneumonia'])
y_pred=pandas.DataFrame(y_pred,columns=['predicted'])
df=pandas.concat([y_test,y_pred],axis=1)

match=[]
for i in range(len(df)):
    if df['pneumonia'][i]==df['predicted'][i]:
        match.append(1)
    else:
        match.append(0)

print('')
print('Radial Kernel')
print('accuracy=', round(mean(match),4))
```



```
###########################################################################################
#FITTING SUPPORT VECTOR BINARY CLASSIFIER WITH SIGMOID KERNEL
svc_sigmoid=SVC(kernel='sigmoid').fit(X_train, y_train)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=svc_sigmoid.predict(X_test)

y_test=pandas.DataFrame(y_test,columns=['pneumonia'])
y_pred=pandas.DataFrame(y_pred,columns=['predicted'])
df=pandas.concat([y_test,y_pred],axis=1)

match=[]
for i in range(len(df)):
    if df['pneumonia'][i]==df['predicted'][i]:
        match.append(1)
    else:
        match.append(0)

print('')
print('Sigmoid Kernel')
print('accuracy=', round(mean(match),4))
```

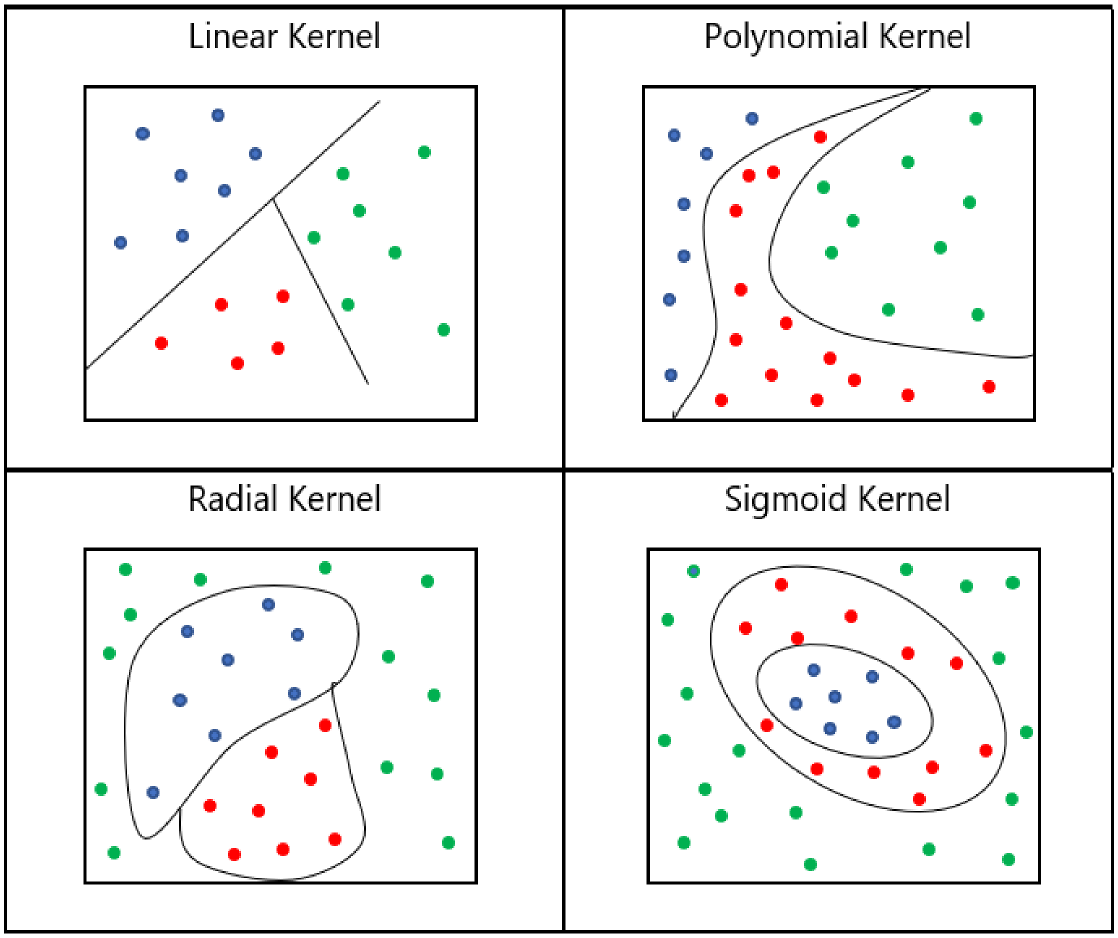
```
Linear Kernel
accuracy= 0.6792

Polynomial Kernel
accuracy= 0.659

Radial Kernel
accuracy= 0.6416

Sigmoid Kernel
accuracy= 0.5491
```

Comparing the accuracies, we conclude that the model with linear kernel has the best fit. □

Support Vector Machine for Multinomial Classifier

One-versus-all is the most practical approach in the case of multinomial classification problems. Suppose there are c classes. The approach dictates the creation of c indicator variables for the classes and running c separate support vector machines for binary classification. For each class, output the predicted probability and choose the class with the highest value as the predicted class.

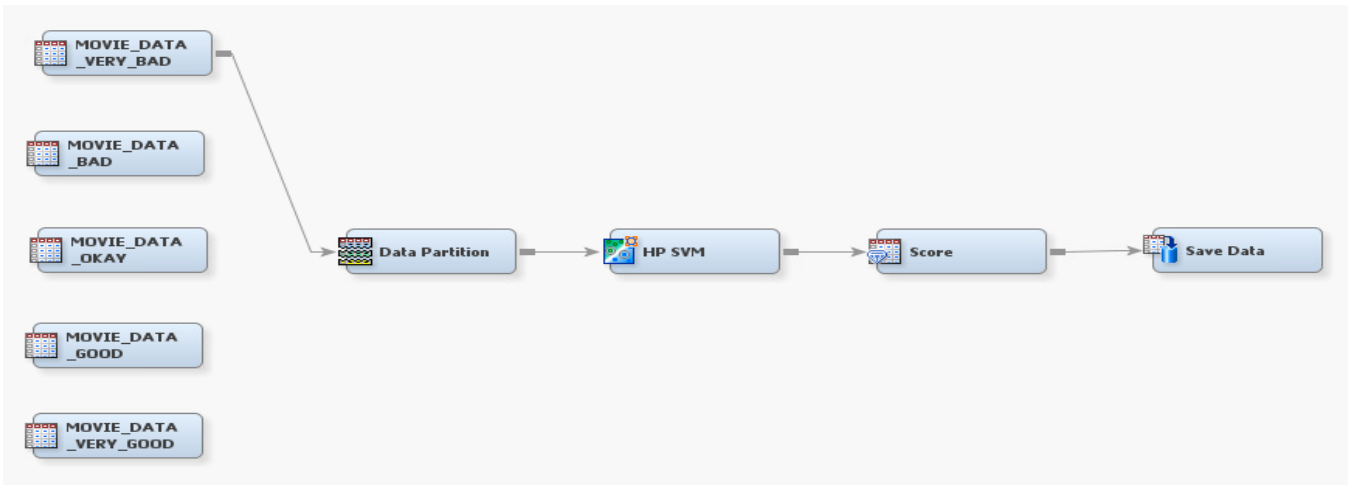
In case of three classes, the separating borders of hyperplanes corresponding to linear, polynomial, radial, and sigmoid kernels are schematically depicted below.

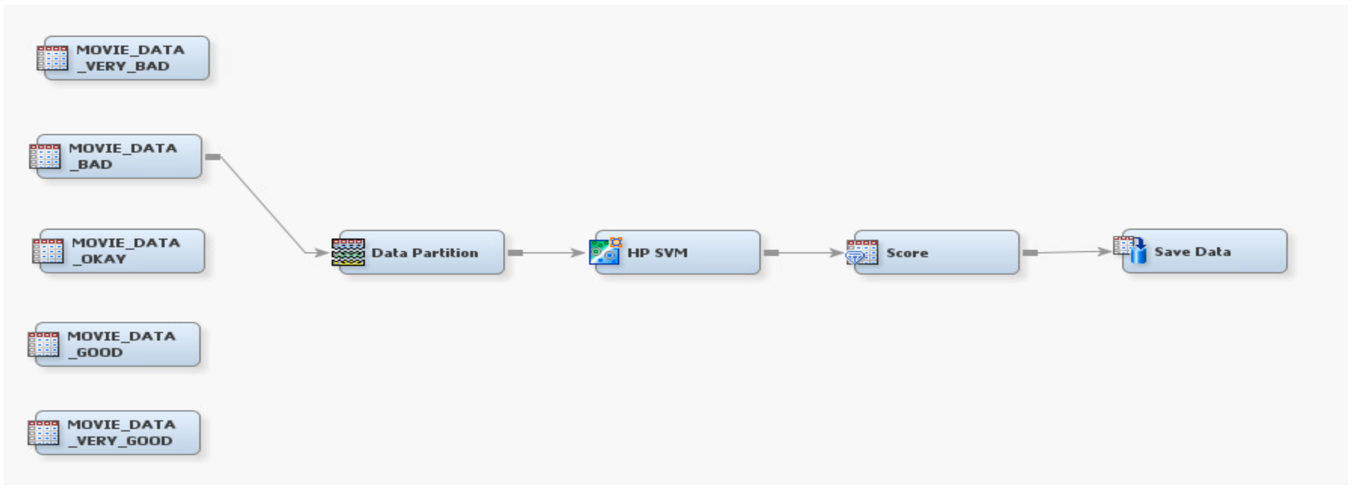


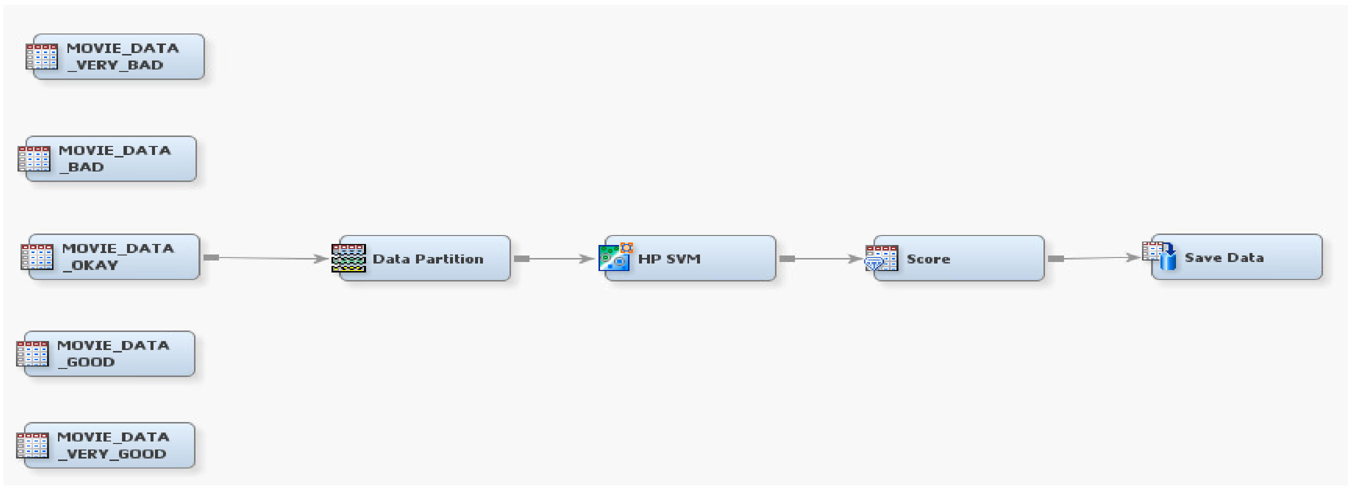
Example. Returning to the data in the file "movie_data.csv", we use SAS Enterprise Miner, R, and Python to fit a support vector multinomial classifier.

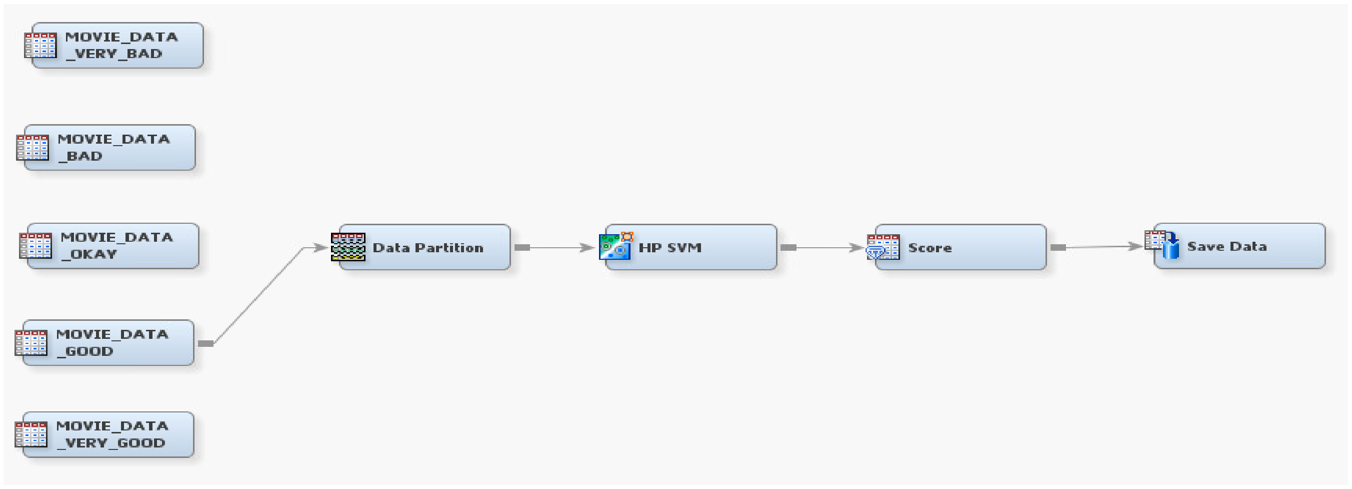
In SAS Enterprise Miner:
We first create indicator variables for the classes "very bad", "bad", "okay", "good", and "very good" and save the data into the file "movie_data_ind.csv", removing the rating variable. Then in Enterprise Miner, we run the given path with polynomial (quadratic), radial, and sigmoid kernels for each of the five classes separately. We specify each indicator variable as the binary target variable and rename the data set to reflect what class is being modeled. We run the five path diagrams

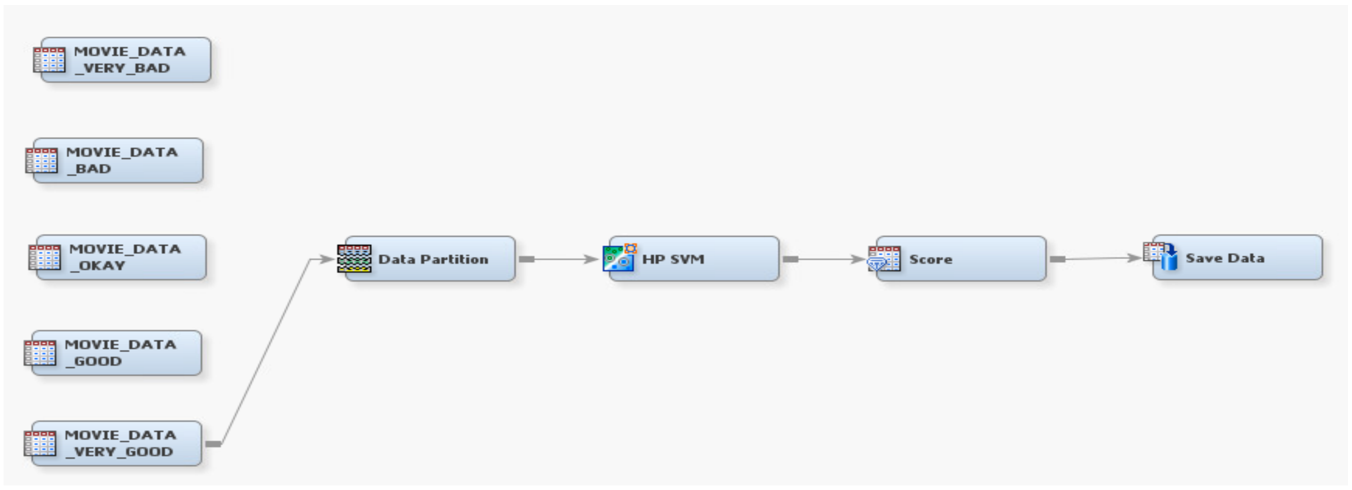
depicted below.






MOVIE_DATA _VERY_BAD
MOVIE_DATA _BAD
MOVIE_DATA _OKAY
Data Partition
HP SVM
Score
Save Data
MOVIE_DATA _GOOD
MOVIE_DATA _VERY_GOOD


MOVIE_DATA _VERY_BAD
MOVIE_DATA _BAD
MOVIE_DATA _OKAY
Data Partition
HP SVM
Score
Save Data
MOVIE_DATA _GOOD
MOVIE_DATA _VERY_GOOD

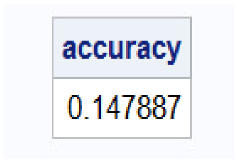


Then we collected all the scored data by type of kernel and identified the class with the highest predicted probability. We then computed and outputted the prediction accuracy for each kernel. The code and output follow.

```
proc import out=sasuser.movies
datafile="./movie_data_ind.csv" dbms=csv replace;
run;

data movies;
set sasuser.movies;
_dataobs_=_N_;
run;

proc sort data=movies;
by _dataobs_;
run;


/*computing prediction accuracy for SVM with quadratic kernel*/
data quadratic_very_bad;
set './quadratic_verybad.sas7bdat';
predprob_very_bad=em_eventprobability;
keep _dataobs_ predprob_very_bad;
run;

proc sort;
by _dataobs_;
```

```
run;

data quadratic_bad;
set './quadratic_bad.sas7bdat';
predprob_bad=em_eventprobability;
keep _dataobs_ predprob_bad;
run;

proc sort;
by _dataobs_;
run;

data quadratic_okay;
set './quadratic_okay.sas7bdat';
predprob_okay=em_eventprobability;
keep _dataobs_ predprob_okay;
run;

proc sort;
by _dataobs_;
run;

data quadratic_good;
set './quadratic_good.sas7bdat';
predprob_good=em_eventprobability;
keep _dataobs_ predprob_good;
run;

proc sort;
by _dataobs_;
run;

data quadratic_very_good;
set './quadratic_verygood.sas7bdat';
predprob_very_good=em_eventprobability;
keep _dataobs_ predprob_very_good;
run;

proc sort;
by _dataobs_;
run;
```

```
data quadratic_kernel;
merge movies quadratic_very_bad quadratic_bad
quadratic_okay quadratic_good quadratic_very_good;
by _dataobs_;
if cmiss(predprob_very_bad, predprob_bad,
predprob_okay, predprob_good, predprob_very_good)=0;
run;

data quadratic_kernel;
set quadratic_kernel;
predprob_max=max(predprob_very_bad, predprob_bad,
predprob_okay, predprob_good, predprob_very_good);
if (predprob_very_good=predprob_max) then pred_class='very good';
if (predprob_very_bad=predprob_max) then pred_class='very bad';
if (predprob_bad=predprob_max) then pred_class='bad';
if (predprob_okay=predprob_max) then pred_class='okay';
if (predprob_good=predprob_max) then pred_class='good';
keep rating pred_class;
run;

data quadratic_kernel;
set quadratic_kernel;
match=(rating=pred_class);
run;

proc sql;
select mean(match) as accuracy
from quadratic_kernel;
quit;
```

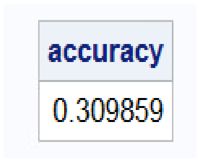


accuracy
0.147887

```
/*******************************************************/
/*computing prediction accuracy for SVM with radial kernel*/
data radial_very_bad;
set './radial_verybad.sas7bdat';
predprob_very_bad=em_eventprobability;
```

```
keep _dataobs_ predprob_very_bad;
run;

proc sort;
by _dataobs_;
run;

data radial_bad;
set './radial_bad.sas7bdat';
predprob_bad=em_eventprobability;
keep _dataobs_ predprob_bad;
run;

proc sort;
by _dataobs_;
run;

data radial_okay;
set './radial_okay.sas7bdat';
predprob_okay=em_eventprobability;
keep _dataobs_ predprob_okay;
run;

proc sort;
by _dataobs_;
run;

data radial_good;
set './radial_good.sas7bdat';
predprob_good=em_eventprobability;
keep _dataobs_ predprob_good;
run;

proc sort;
by _dataobs_;
run;

data radial_very_good;
set './radial_verygood.sas7bdat';
predprob_very_good=em_eventprobability;
keep _dataobs_ predprob_very_good;
run;
```

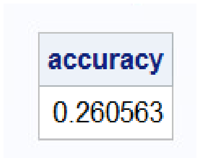
```
proc sort;
by _dataobs_;
run;

data radial_kernel;
merge movies radial_very_bad radial_bad
radial_okay radial_good radial_very_good;
by _dataobs_;
if cmiss(predprob_very_bad, predprob_bad,
predprob_okay, predprob_good, predprob_very_good)=0;
run;

data radial_kernel;
set radial_kernel;
predprob_max=max(predprob_very_bad, predprob_bad,
predprob_okay, predprob_good, predprob_very_good);
if (predprob_very_good=predprob_max) then pred_class='very good';
if (predprob_very_bad=predprob_max) then pred_class='very bad';
if (predprob_bad=predprob_max) then pred_class='bad';
if (predprob_okay=predprob_max) then pred_class='okay';
if (predprob_good=predprob_max) then pred_class='good';
keep rating pred_class;
run;

data radial_kernel;
set radial_kernel;
match=(rating=pred_class);
run;

proc sql;
select mean(match) as accuracy
from radial_kernel;
quit;
```



accuracy
0.309859

```
/*******************************************************/
/*computing prediction accuracy for SVM with sigmoid kernel*/
data sigmoid_very_bad;
set './sigmoid_verybad.sas7bdat';
predprob_very_bad=em_eventprobability;
keep _dataobs_ predprob_very_bad;
run;

proc sort;
by _dataobs_;
run;

data sigmoid_bad;
set './sigmoid_bad.sas7bdat';
predprob_bad=em_eventprobability;
keep _dataobs_ predprob_bad;
run;

proc sort;
by _dataobs_;
run;

data sigmoid_okay;
set './sigmoid_okay.sas7bdat';
predprob_okay=em_eventprobability;
*keep _dataobs_ predprob_okay;
run;

proc sort;
by _dataobs_;
run;

data sigmoid_good;
set './sigmoid_good.sas7bdat';
predprob_good=em_eventprobability;
keep _dataobs_ predprob_good;
run;

proc sort;
by _dataobs_;
run;
```

```
data sigmoid_very_good;
set './sigmoid_verygood.sas7bdat';
predprob_very_good=em_eventprobability;
keep _dataobs_ predprob_very_good;
run;

proc sort;
by _dataobs_;
run;

data sigmoid_kernel;
merge movies sigmoid_very_bad sigmoid_bad
sigmoid_okay sigmoid_good sigmoid_very_good;
by _dataobs_;
if cmiss(predprob_very_bad, predprob_bad,
predprob_okay, predprob_good, predprob_very_good)=0;
run;

data sigmoid_kernel;
set sigmoid_kernel;
predprob_max=max(predprob_very_bad, predprob_bad,
predprob_okay, predprob_good, predprob_very_good);
if (predprob_very_good=predprob_max) then pred_class='very good';
if (predprob_very_bad=predprob_max) then pred_class='very bad';
if (predprob_bad=predprob_max) then pred_class='bad';
if (predprob_okay=predprob_max) then pred_class='okay';
if (predprob_good=predprob_max) then pred_class='good';
keep rating pred_class;
run;

data sigmoid_kernel;
set sigmoid_kernel;
match=(rating=pred_class);
run;

proc sql;
select mean(match) as accuracy
from sigmoid_kernel;
quit;
```



accuracy
0.260563

We see that the largest accuracy is for the model with the radial kernel.

In R:

```
movie.data<- read.csv(file="./movie_data.csv", header=TRUE, sep=",")

movie.data$gender<- ifelse(movie.data$gender=='M',1,0)
movie.data$member<- ifelse(movie.data$member=='yes',1,0)
#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
set.seed(444625)
sample <- sample(c(TRUE, FALSE), nrow(movie.data), replace=TRUE, prob=c(0.8,0.2))
train<- movie.data[sample,]
test<- movie.data[!sample,]

train.x<- data.matrix(train[-5])
train.y<- data.matrix(train[5])
test.x<- data.matrix(test[-5])
test.y<- data.matrix(test[5])

library(e1071)

#FITTING SVM WITH LINEAR KERNEL
svm.multiclass<- svm(as.factor(rating) ~ age + gender + member + nmovies,
data=train, kernel="linear")

#computing prediction accuracy for testing data
pred.y<- as.numeric(predict(svm.multiclass, test.x))

print(paste("accuracy=", round(1-mean(test.y!=pred.y),digits=4)))
```

```
"accuracy= 0.2892"
```

```
#FITTING SVM WITH POLYNOMIAL KERNEL
svm.multiclass<- svm(as.factor(rating) ~ age + gender + member + nmovies,
data=train, kernel="polynomial")

#computing prediction accuracy for testing data
pred.y<- as.numeric(predict(svm.multiclass, test.x))

print(paste("accuracy=", round(1-mean(test.y!=pred.y),digits=4)))
```

```
"accuracy= 0.3133"
```

```
#FITTING SVM WITH RADIAL KERNEL
svm.multiclass<- svm(as.factor(rating) ~ age + gender + member + nmovies,
data=train, kernel="radial")

#computing prediction accuracy for testing data
pred.y<- as.numeric(predict(svm.multiclass, test.x))

print(paste("accuracy=", round(1-mean(test.y!=pred.y),digits=4)))
```

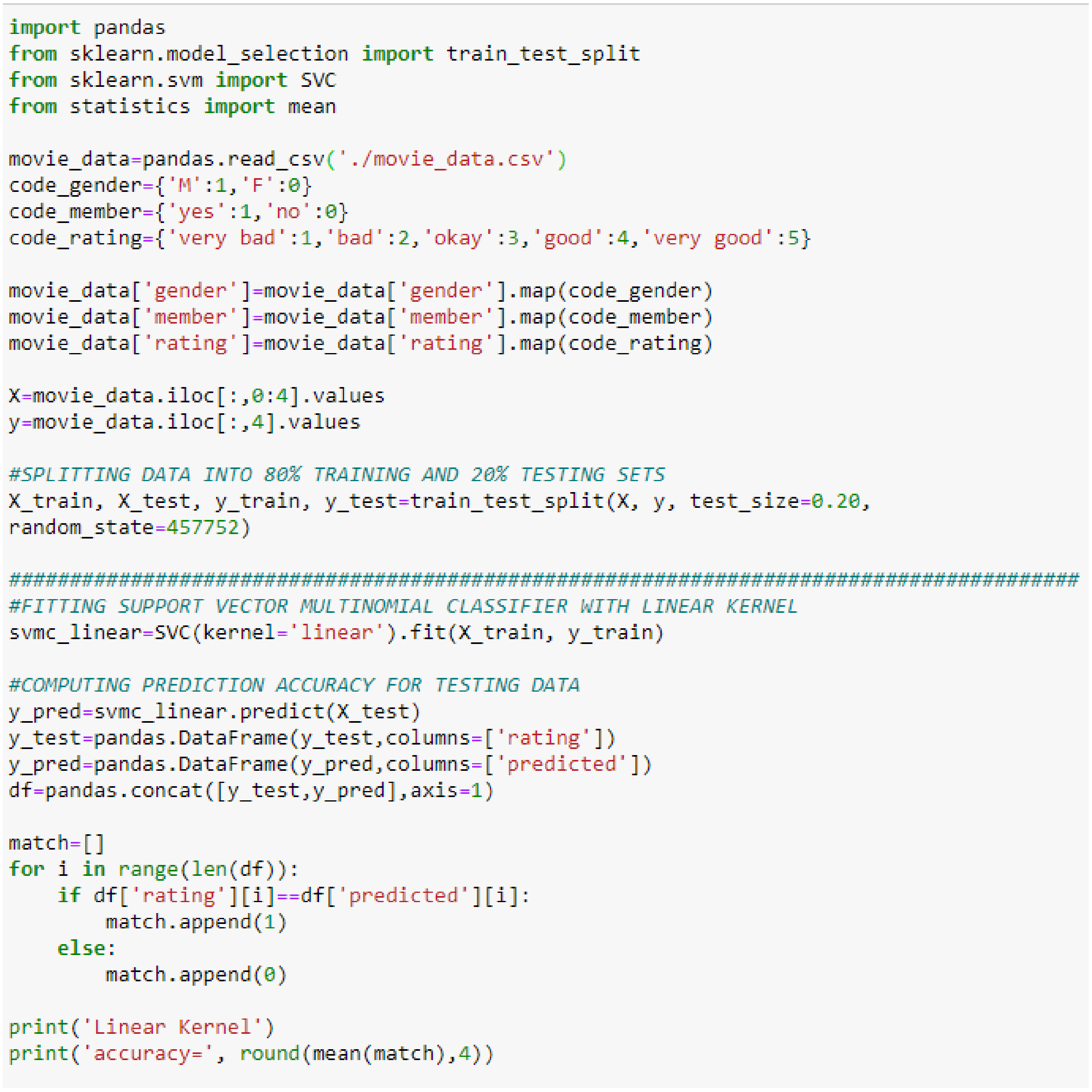
```
"accuracy= 0.3133"
```

```
#FITTING SVM WITH SIGMOID KERNEL
svm.multiclass<- svm(as.factor(rating) ~ age + gender + member + nmovies,
data=train, kernel="sigmoid")

#computing prediction accuracy for testing data
pred.y<- as.numeric(predict(svm.multiclass, test.x))

print(paste("accuracy=", round(1-mean(test.y!=pred.y),digits=4)))
```

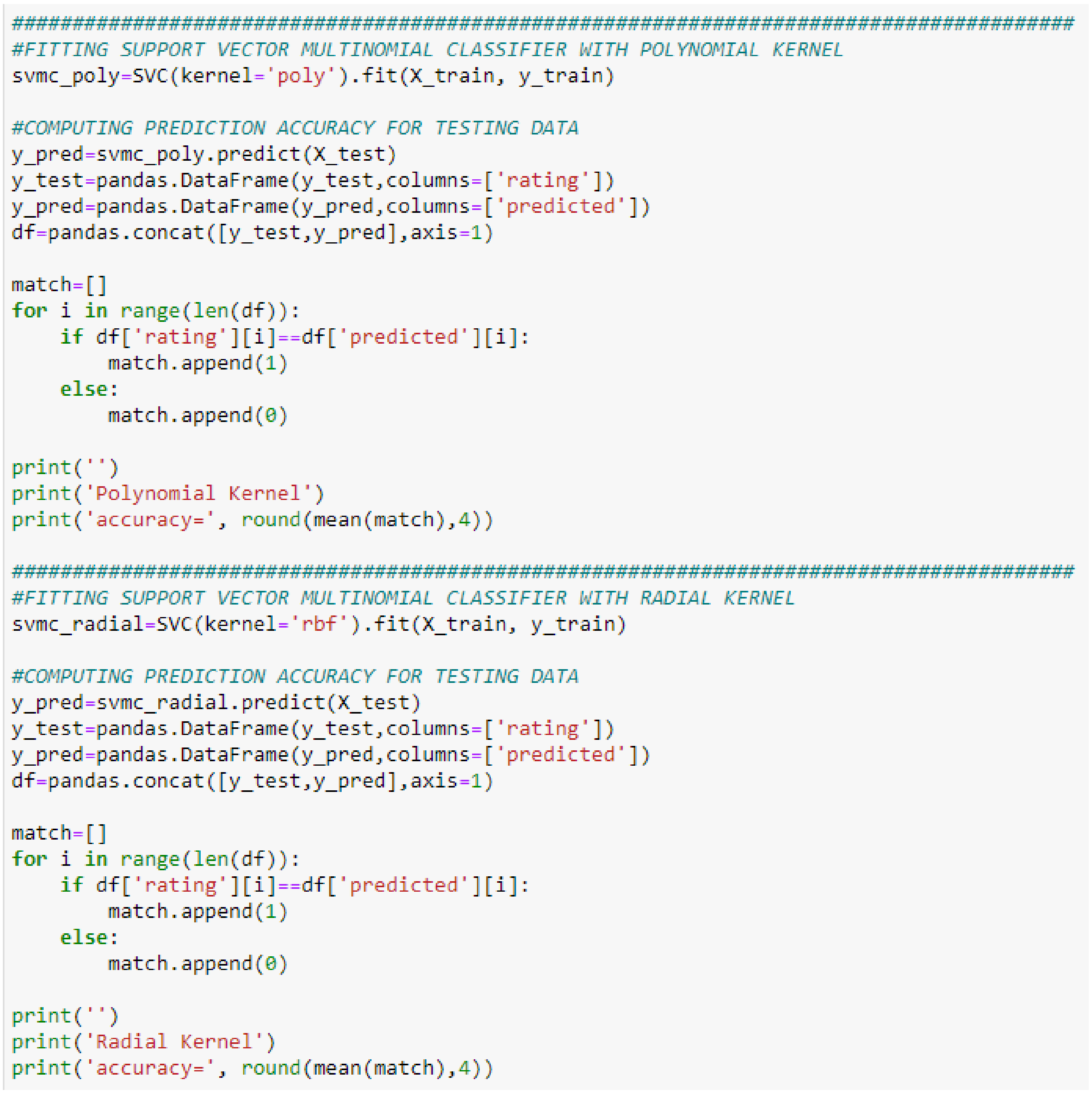
```
"accuracy= 0.2651"
```

The models with polynomial and radial kernels have the best fit.

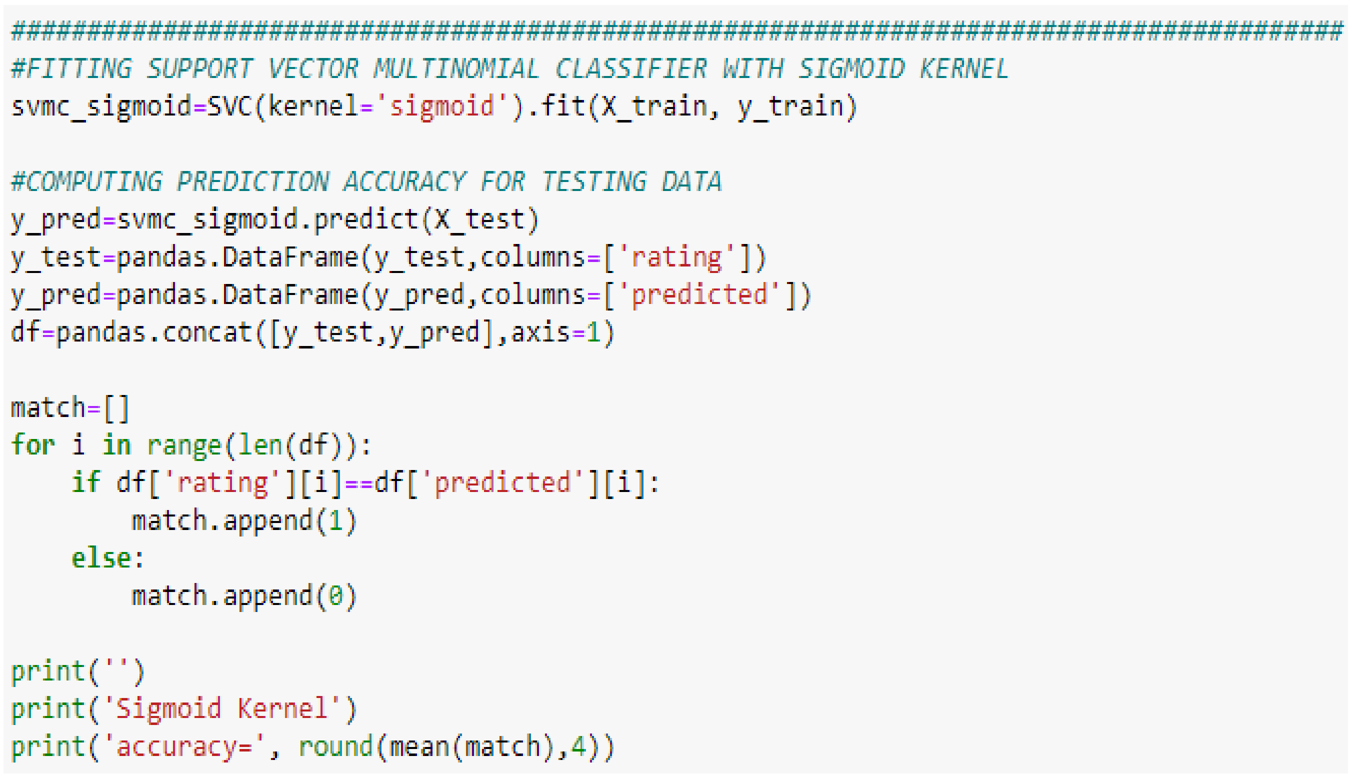
In Python:



```
import pandas
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from statistics import mean

movie_data=pandas.read_csv('./movie_data.csv')
code_gender={'M':1,'F':0}
code_member={'yes':1,'no':0}
code_rating={'very bad':1,'bad':2,'okay':3,'good':4,'very good':5}

movie_data['gender']=movie_data['gender'].map(code_gender)
movie_data['member']=movie_data['member'].map(code_member)
movie_data['rating']=movie_data['rating'].map(code_rating)

X=movie_data.iloc[:,0:4].values
y=movie_data.iloc[:,4].values

#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
X_train, X_test, y_train, y_test=train_test_split(X, y, test_size=0.20,
random_state=457752)

########################################################################################
#FITTING SUPPORT VECTOR MULTINOMIAL CLASSIFIER WITH LINEAR KERNEL
svmc_linear=SVC(kernel='linear').fit(X_train, y_train)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=svmc_linear.predict(X_test)
y_test=pandas.DataFrame(y_test,columns=['rating'])
y_pred=pandas.DataFrame(y_pred,columns=['predicted'])
df=pandas.concat([y_test,y_pred],axis=1)

match=[]
for i in range(len(df)):
    if df['rating'][i]==df['predicted'][i]:
        match.append(1)
    else:
        match.append(0)

print('Linear Kernel')
print('accuracy=', round(mean(match),4))
```



```
##########################################################################################
#FITTING SUPPORT VECTOR MULTINOMIAL CLASSIFIER WITH POLYNOMIAL KERNEL
svmc_poly=SVC(kernel='poly').fit(X_train, y_train)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=svmc_poly.predict(X_test)
y_test=pandas.DataFrame(y_test,columns=['rating'])
y_pred=pandas.DataFrame(y_pred,columns=['predicted'])
df=pandas.concat([y_test,y_pred],axis=1)

match=[]
for i in range(len(df)):
    if df['rating'][i]==df['predicted'][i]:
        match.append(1)
    else:
        match.append(0)

print('')
print('Polynomial Kernel')
print('accuracy=', round(mean(match),4))

##########################################################################################
#FITTING SUPPORT VECTOR MULTINOMIAL CLASSIFIER WITH RADIAL KERNEL
svmc_radial=SVC(kernel='rbf').fit(X_train, y_train)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=svmc_radial.predict(X_test)
y_test=pandas.DataFrame(y_test,columns=['rating'])
y_pred=pandas.DataFrame(y_pred,columns=['predicted'])
df=pandas.concat([y_test,y_pred],axis=1)

match=[]
for i in range(len(df)):
    if df['rating'][i]==df['predicted'][i]:
        match.append(1)
    else:
        match.append(0)

print('')
print('Radial Kernel')
print('accuracy=', round(mean(match),4))
```



```python
###########################################################################################
#FITTING SUPPORT VECTOR MULTINOMIAL CLASSIFIER WITH SIGMOID KERNEL
svmc_sigmoid=SVC(kernel='sigmoid').fit(X_train, y_train)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=svmc_sigmoid.predict(X_test)
y_test=pandas.DataFrame(y_test,columns=['rating'])
y_pred=pandas.DataFrame(y_pred,columns=['predicted'])
df=pandas.concat([y_test,y_pred],axis=1)

match=[]
for i in range(len(df)):
    if df['rating'][i]==df['predicted'][i]:
        match.append(1)
    else:
        match.append(0)

print('')
print('Sigmoid Kernel')
print('accuracy=', round(mean(match),4))
```

```
Linear Kernel
accuracy= 0.3421

Polynomial Kernel
accuracy= 0.3224

Radial Kernel
accuracy= 0.3553

Sigmoid Kernel
accuracy= 0.3289
```

The model with radial kernel has the best fit. □.