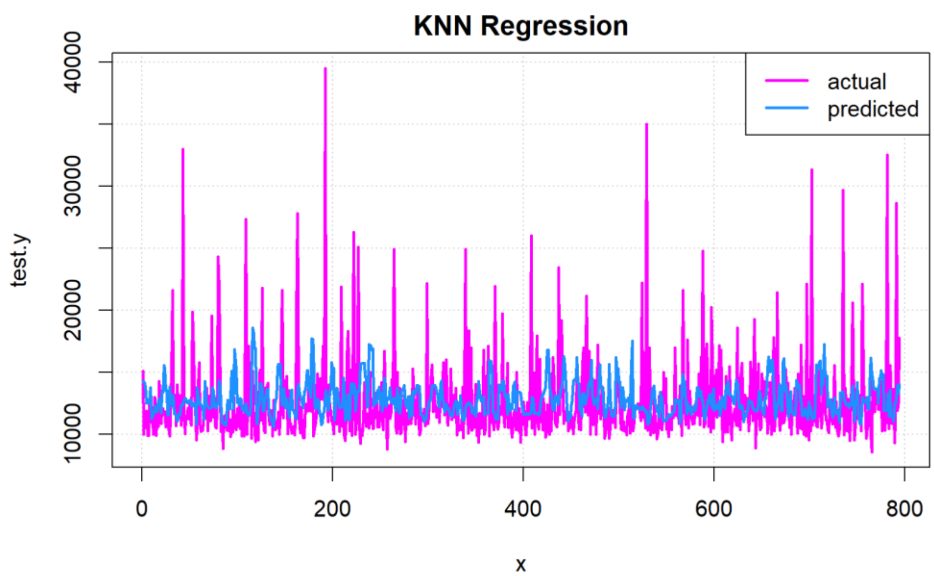
STAT 574 Midterm

Derrick Edwards

Problem 1. Use the data in the file “hospital_data.csv” to fit a k-nearest neighbor regression. Compute prediction accuracy within 10%, 15%, and 20% of the actual values. Plot the actual and predicted values in the same coordinate system.

R Output

```
[1] 0.3513854
[1] 0.5151134
[1] 0.6700252
```



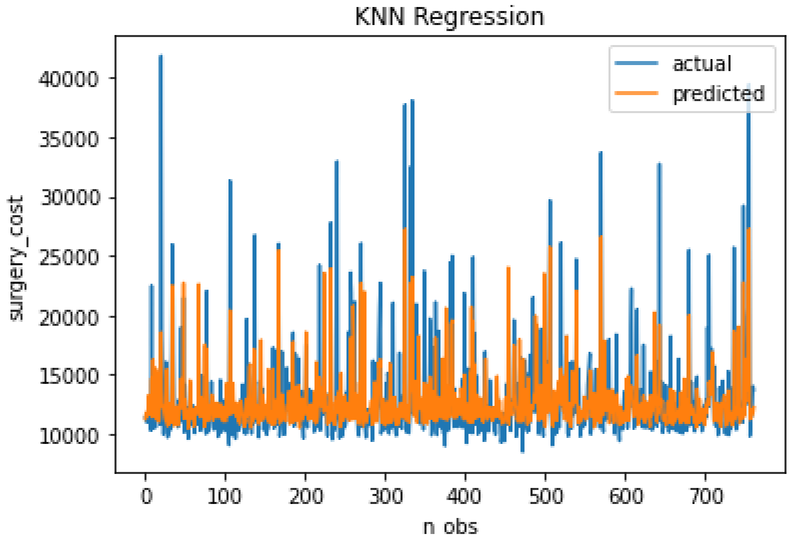
R Code

```
# Problem 1
# k-nearest neighborhood regression - compute prediction accuracy 10,15, 20%
# plot acutal and predicted values in same coordinate system

hospital <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/HW1STAT574S23/DATA
SETS/hospital_data.csv")

# split data 80% train 20% test
set.seed(1094543)
sample <- sample(c(TRUE, FALSE), nrow(hospital), replace=TRUE, prob = c(0.8,0.2))
train <- hospital[sample,]
test <- hospital[!sample,]
```

```
View(hospital)

train.x<- data.matrix(train[-7])
train.y<- data.matrix(train[7])
test.x<- data.matrix(test[-7])
test.y<- data.matrix(test[7])



#TRAINING K-NEAREST NEIGHBOR REGRESSION
library(caret)
print(train(surgery_cost ~ gender + age + ASA + BMI + surgery_duration_min,
        data=train, method="knn"))

#FITTING OPTIMAL KNN REGRESSION (K=9)
knn.reg<- knnreg(train.x, train.y, k=9)

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


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

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

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

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

#PLOTTING ACTUAL AND RPEDICTED VALUES FOR TESTING DATA
x<- 1:length(test.y)
plot(x, test.y, type="l", lwd=2, col="magenta", main="KNN Regression",
panel.first=grid())
lines(x, pred.y, lwd=2, col="dodgerblue")
legend("topright", c("actual", "predicted"), lty=1, lwd=2,
col=c("magenta","dodgerblue"))
```

Python Output

```
accuracy within 10% = 0.5118
accuracy within 15% = 0.7021
accuracy within 20% = 0.8084
```



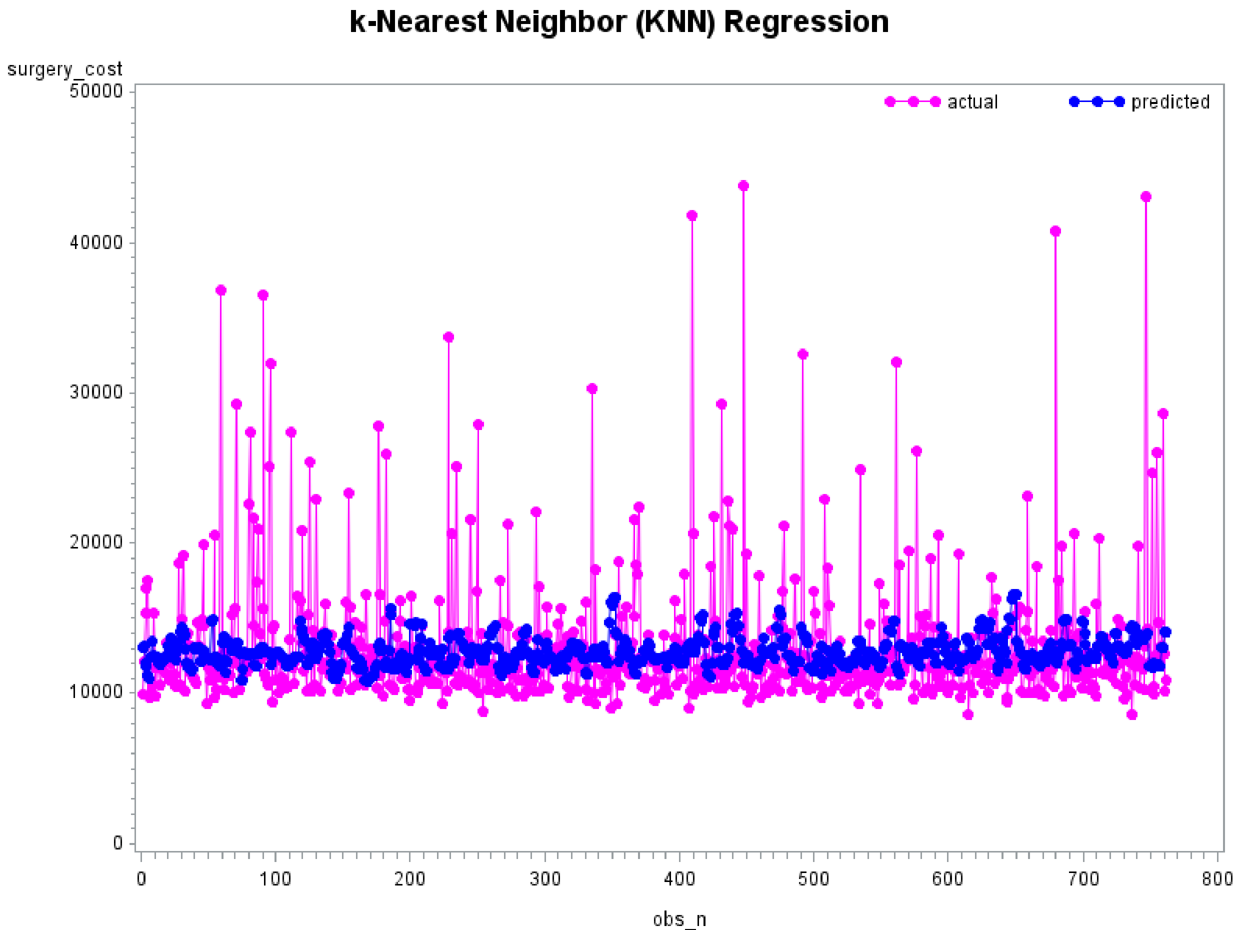
Python Code

```
import pandas
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from statistics import mean

# read in data
hospital=pandas.read_csv(r'C:\Users\saedw\OneDrive\Desktop\STAT 574 Data
Mining\HW1STAT574S23\DATA SETS\hospital_data.csv')
coding={'M': 1, 'F': 0}
hospital['gender']=hospital['gender'].map(coding)

X=hospital.iloc[:,1:6].values
y=hospital.iloc[:,6].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=348644)

#FITTING kNN REGRESSION
reg=KNeighborsRegressor(n_neighbors=63)
kNN_reg=reg.fit(X_train, y_train)


#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
y_pred=kNN_reg.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)

#accuracy within 10%
accuracy10=mean(ind10)
print('accuracy within 10% =', round(accuracy10,4))

#accuracy within 15%
accuracy15=mean(ind15)
print('accuracy within 15% =', round(accuracy15,4))

#accuracy within 20%
accuracy20=mean(ind20)
print('accuracy within 20% =', round(accuracy20,4))


#plotting actual and predicted obsevations vs. observation number
import matplotlib.pyplot as plt

n_obs=list(range(0,len(y_test)))
plt.plot(n_obs, y_test, label="actual")
plt.plot(n_obs, y_pred, label="predicted")
plt.xlabel('n_obs')
plt.ylabel('surgery_cost')
plt.title('KNN Regression')
plt.legend()
plt.show()
```

SAS Output

accuracy10	accuracy15	accuracy20
0.354331	0.519685	0.649606



SAS Code

```
proc import out=sasuser.hospital
file="\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\Data\hospital_data.csv"
dbms=csv replace;
run;

/*Running Memory Based Reasoning (MBR)(or kNN regression)
in Enterprise Miner*/
libname midp1 "\\vdi-
fileshare01\UEMprofiles\017365554\Desktop\KNNReg\Workspaces\EMWS1\emsave";
/*COMPUTING ACCURACY WITHIN 10%, 15%, AND 20%*/
data accuracy;
set midp1.em_save_test;
ind10=(abs(R_surgery_cost)<0.10*surgery_cost);
ind15=(abs(R_surgery_cost)<0.15*surgery_cost);
ind20=(abs(R_surgery_cost)<0.20*surgery_cost);
obs_n=_N_;
run;

proc sql;
select mean(ind10) as accuracy10,
mean(ind15) as accuracy15, mean(ind20) as
accuracy20
```

```
from accuracy;
quit;

/*PLOTTING ACTUAL AND PREDICTED VALUES FOR TESTING DATA*/;
goptions reset=all border;
title1 "k-Nearest Neighbor (KNN) Regression";
symbol1 interpol=join value=dot color=magenta;
symbol2 interpol=join value=dot color=blue;
legend1 value=("actual" "predicted")
position=(top right inside) label=none;
proc gplot data=accuracy;
plot surgery_cost*obs_n
EM_PREDICTION*obs_n/ overlay legend=legend1;
run;
```

Problem 2. Use the data in the file "card_transdata.csv" to fit a k-nearest neighbor binary classifier with kk = 9. Compute prediction accuracy.

R Output

```
[1] "accuracy= 0.9321"
[1] "accuracy= 0.9321"
```

R Code

```
# Problem 2
# fit k-nearest neighbor binary classifier
# compute prediction accuracy

credit_data <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/HW1STAT574S23/DATA SETS/card_transdata.csv")


View(credit_data)

# split data 80% train 20% test
set.seed(6749379)
sample <- sample(c(TRUE,FALSE), nrow(credit_data), replace=TRUE,
        prob=c(0.8,0.2))
train <- credit_data[sample,]
test <- credit_data[!sample,]

train.x<- data.matrix(train[-8])
train.y<- data.matrix(train[8])
test.x<- data.matrix(test[-8])
test.y<- data.matrix(test[8])
```

```
#TRAINING K-NEAREST NEIGHBOR BINARY CLASSIFIER
library(caret)
print(train(as.factor(fraud)~., data=train, method="knn"))

#FITTING OPTIMAL KNN BINARY CLASSIFIER (K=9)
knn.class<- knnreg(train.x, train.y, k=9)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
pred.prob<- predict(knn.class, test.x)

len<- length(pred.prob)
pred.y<- c()
match<- c()
for (i in 1:len){
  pred.y[i]<- ifelse(pred.prob[i]>=0.5, 1,0)
  match[i]<- ifelse(test.y[i]==pred.y[i], 1,0)
}
print(paste("accuracy=",round(mean(match),digits=4)))

#alternative (frugal) way
pred.y1<- floor(0.5+predict(knn.class, test.x))
print(paste("accuracy=", round(1-mean(test.y!=pred.y1),digits=4)))
```

Python Output

```
Accuracy:  0.91
```

Python Code

```python
import pandas
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from statistics import mean

card_data=pandas.read_csv(r'C:\Users\saedw\OneDrive\Desktop\STAT 574 Data
Mining\HW1STAT574S23\DATA SETS\card_transdata.csv')
X=card_data.iloc[:,0:7].values
y=card_data.iloc[:,7]

#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=786756)

#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=459147)
```

```python
#FITTING kNN BINARY CLASSIFIER WITH k=4
biclass=KNeighborsClassifier(n_neighbors=87)
kNN_biclass=biclass.fit(X_train, y_train)

# print accuracy score
from sklearn.metrics import accuracy_score
# store predicted values from testing set
y_pred=kNN_biclass.predict(X_test)

accuracy=accuracy_score(y_test, y_pred)

print("Accuracy: ", accuracy)
```

SAS Output

accuracy
1

SAS Code

```sas
proc import out=sasuser.card_data
datafile="\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT 
574\Data\card_transdata.csv"
dbms=csv replace;
run;

/*Running Memory Based Reasoning (MBR)(or kNN binary classifier) 
in Enterprise Miner*/

/*COMPUTING PREDICTION ACCURACY*/
libname midp2 "\\vdi-
fileshare01\UEMprofiles\017365554\Desktop\KnnBin\Workspaces\EMWS1\emsave";

data accuracy;
set midp2.em_save_test;
match=(em_classification=em_classtarget);
run;

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

Problem 3. Use the data in the file “concussions_data.csv” to fit a k-nearest neighbor multinomial classifier. Compute prediction accuracy.

R Output

```
[1] "accuracy= 0.8404"
```

R Code

```
# Problem 3
# k-nearest neighbor multinomial classification
# compute prediction accuracy

concuss <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/HW1STAT574S23/DATA SETS/concussions_data.csv")

# split data 80% train 20% test
set.seed(898323)
sample <- sample(c(TRUE,FALSE), nrow(concuss), replace=TRUE,
        prob=c(0.8,0.2))
train <- concuss[sample,]
test <- concuss[!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])


#FITTING K-NEAREST NEIGHBOR MULTINOMIAL CLASSIFIER
#k=3 reasonably maximizes prediction accuracy for testing set
library(caret)
knn.mclass<- knnreg(train.x, train.y, k=3)

#COMPUTING PREDICTION ACCURACY FOR TESTING DATA
pred.y<- round(predict(knn.mclass, test.x), digits=0)
print(paste("accuracy=", round(1-mean(test.y!=pred.y),digits=4)))
```

Python Output

```
Accuracy:  0.819047619047619
```

Python Code

```
# Problem 3 - k-nearest neighbor multinomial classification
    # print accuracy score

import pandas
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from statistics import mean
```

```python
concussion_data=pandas.read_csv(r'C:\Users\saedw\OneDrive\Desktop\STAT 574 Data Mining\HW1STAT574S23\DATA SETS\concussions_data.csv')

code_position={'Offensive Lineman': 1, 'Cornerback': 2, 'Running Back': 3,'Wide Receiver': 4,
'Quarterback': 5}
code_concussion={'mild': 1, 'moderate': 2, 'severe': 3}

concussion_data['position']=concussion_data['position'].map(code_position)
concussion_data['concussion']=concussion_data['concussion'].map(code_concussion)


X=concussion_data.iloc[:,0:4]
y=concussion_data.iloc[:,4]

#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=599555)

#FITTING kNN MULTINOMIAL CLASSIFIER
multiclass=KNeighborsClassifier(n_neighbors=31)
kNN_multiclass=multiclass.fit(X_train, y_train)

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

accuracy=accuracy_score(y_test, y_pred)

print("Accuracy: ", accuracy)
```

SAS Output

accuracy
0.663551

SAS Code

```sas
proc import out=sasuser.concuss
datafile="\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT 574\Data\concussions_data.csv"
dbms=csv replace;
run;

/*Running Memory Based Reasoning (MBR)(or kNN binary classifier) in Enterprise Miner*/

/*COMPUTING PREDICTION ACCURACY*/
```

```
libname midp2 "\\vdi-
fileshare01\UEMprofiles\017365554\Desktop\KnnMulti\Workspaces\EMWS1\emsave";

data accuracy;
set midp2.em_save_test;
match=(em_classification=em_classtarget);
run;

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

Problem 4. Use the data in the file "hospital_data.csv" to fit a support vector regression with linear, polynomial, radial, and sigmoid kernels. Compute prediction accuracy within 10%, 15%, and 20% of the actual values. Choose the best-fitted model. Use R and Python only.

R Output

```
[1] "Linear Kernel"
[1] "within 10%: 0.5315"
[1] "within 15%: 0.7015"
[1] "within 20%: 0.8111"
[1] "Polynomial Kernel"
[1] "within 10%: 0.5768"
[1] "within 15%: 0.7506"
[1] "within 20%: 0.8539"
[1] "Radial Kernel"
[1] "within 10%: 0.5781"
[1] "within 15%: 0.7355"
[1] "within 20%: 0.8401"
[1] "Sigmoid Kernel"
[1] "within 10%: 0.029"
[1] "within 15%: 0.0378"
[1] "within 20%: 0.0453"
```

R Code

```
# Problem 4
# fit support vector regression with linear, polynomial, radial, and sigmoid
 # kernals - compute prediction accuracy 10, 15, 20% - choose best fitted model

hospital <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/HW1STAT574S23/DATA SETS/hospital_data.csv")


# split data 80% train 20% test
set.seed(1094543)
```

```
sample <- sample(c(TRUE, FALSE), nrow(hospital), replace=TRUE, prob = c(0.8,0.2))
train <- hospital[sample,]
test <- hospital[!sample,]

test.x<- data.matrix(test[-7])
test.y<- data.matrix(test[7])


library(e1071)

#FITTING SVR WITH LINEAR KERNEL ########################################
svm.reg<- svm(surgery_cost ~ age + gender + BMI + ASA + surgery_duration_min,
        data=train, kernel="linear")

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

#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')
print(paste('within 10%:', round(mean(accuracy10),4)))
print(paste('within 15%:', round(mean(accuracy15),4)))
print(paste('within 20%:', round(mean(accuracy20),4)))


#FITTING SVR WITH POLYNOMIAL KERNEL ########################################
svm.reg<- svm(surgery_cost ~ age + gender + BMI + ASA + surgery_duration_min,
        data=train, kernel="poly")

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

#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')
print(paste('within 10%:', round(mean(accuracy10),4)))
print(paste('within 15%:', round(mean(accuracy15),4)))
print(paste('within 20%:', round(mean(accuracy20),4)))

#FITTING SVR WITH RADIAL KERNEL ########################################
svm.reg<- svm(surgery_cost ~ age + gender + BMI + ASA + surgery_duration_min,
        data=train, kernel="radial")

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

#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')
print(paste('within 10%:', round(mean(accuracy10),4)))
print(paste('within 15%:', round(mean(accuracy15),4)))
print(paste('within 20%:', round(mean(accuracy20),4)))

#FITTING SVR WITH SIGMOID KERNEL ######################################
svm.reg<- svm(surgery_cost ~ age + gender + BMI + ASA + surgery_duration_min,
        data=train, kernel="sigmoid")

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

#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')
print(paste('within 10%:', round(mean(accuracy10),4)))
print(paste('within 15%:', round(mean(accuracy15),4)))
print(paste('within 20%:', round(mean(accuracy20),4)))

Python Output

```
Linear Kernel
within 10%: 0.4948
within 15%: 0.6798
within 20%: 0.8084

Polynomial Kernel
within 10%: 0.4856
within 15%: 0.7139
within 20%: 0.8333

Radial Kernel
within 10%: 0.4816
within 15%: 0.6798
within 20%: 0.811

Sigmoid Kernel
within 10%: 0.4869
within 15%: 0.6732
within 20%: 0.8018
```

Python Code

```
# Problem 4 - support vector regression
    # fit kernals: linear, polynomial, radial, sigmoid
    # compute accuracies 10,15,20% - choose best model

import pandas
from sklearn.model_selection import train_test_split
from sklearn.svm import SVR

# read in data
hospital=pandas.read_csv(r'C:\Users\saedw\OneDrive\Desktop\STAT 574 Data Mining\HW1STAT574S23\DATA SETS\hospital_data.csv')
coding={'M': 1, 'F': 0}
hospital['gender']=hospital['gender'].map(coding)

X=hospital.iloc[:,1:6].values
y=hospital.iloc[:,6].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=348644)

################################################################################
#######
#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)

################################################################################
#######
#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')
#accuracy within 10%
accuracy10=sum(ind10)/len(ind10)
print('within 10%:', round(accuracy10,4))

#accuracy within 15%
accuracy15=sum(ind15)/len(ind15)
print('within 15%:', round(accuracy15,4))

#accuracy within 20%
accuracy20=sum(ind20)/len(ind20)
print('within 20%:', round(accuracy20,4))

################################################################################
#######
#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')
#accuracy within 10%
accuracy10=sum(ind10)/len(ind10)
print('within 10%:', round(accuracy10,4))

#accuracy within 15%
accuracy15=sum(ind15)/len(ind15)
print('within 15%:', round(accuracy15,4))

#accuracy within 20%
accuracy20=sum(ind20)/len(ind20)
print('within 20%:', round(accuracy20,4))


#################################################################################
#######
#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')
#accuracy within 10%
accuracy10=sum(ind10)/len(ind10)
print('within 10%:', round(accuracy10,4))

#accuracy within 15%
accuracy15=sum(ind15)/len(ind15)
print('within 15%:', round(accuracy15,4))

#accuracy within 20%
accuracy20=sum(ind20)/len(ind20)
print('within 20%:', round(accuracy20,4))

#################################################################################
#######
#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')
#accuracy within 10%
accuracy10=sum(ind10)/len(ind10)
print('within 10%:', round(accuracy10,4))

#accuracy within 15%
accuracy15=sum(ind15)/len(ind15)
print('within 15%:', round(accuracy15,4))

#accuracy within 20%
accuracy20=sum(ind20)/len(ind20)
print('within 20%:', round(accuracy20,4))
```

Problem 5. Use the data in the file “card_transdata.csv” to fit a support vector binary classifier. Specify linear, polynomial, radial, and sigmoid kernels (whichever are possible to fit). Compute and compare prediction accuracies.

R Output

```
[1] "accuracy= 0.9755"
[1] "accuracy= 0.9755"
[1] "accuracy= 0.9828"
[1] "accuracy= 0.9338"
```

R Code

```
# Problem 5
# support vector binary classifier with linear, polynomial, radial, and sigmoid
  # kernals and compute prediction accuracy


credit_data <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data
Mining/HW1STAT574S23/DATA SETS/card_transdata.csv")

View(credit_data)

# split data 80% train 20% test
set.seed(482044)
sample <- sample(c(TRUE, FALSE), nrow(credit_data), replace=TRUE,
             prob = c(0.8,0.2))
train <- credit_data[sample,]
test <- credit_data[!sample,]

train.x<- data.matrix(train[-8])
train.y<- data.matrix(train[8])
test.x<- data.matrix(test[-8])
test.y<- data.matrix(test[8])

library(e1071)


#FITTING SVM WITH LINEAR KERNEL ##############################################
svm.class<- svm(as.factor(fraud) ~ ., data=train, kernel="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)))


#FITTING SVM WITH Polynomial KERNEL ##############################################
svm.class<- svm(as.factor(fraud) ~ ., data=train, kernel="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)))

#FITTING SVM WITH radial KERNEL ##############################################
svm.class<- svm(as.factor(fraud) ~ ., data=train, kernel="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)))

#FITTING SVM WITH sigmoid KERNEL ##############################################
svm.class<- svm(as.factor(fraud) ~ ., data=train, kernel="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)))
```

Python Output

```
Linear Kernal Accuracy:  0.9675
Polynomial Kernal Accuracy:  0.9225
Radial Kernal Accuracy:  0.92
Sigmoid Kernal Accuracy:  0.905
```

Python Code

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

card_data=pandas.read_csv(r'C:\Users\saedw\OneDrive\Desktop\STAT 574 Data
Mining\HW1STAT574S23\DATA SETS\card_transdata.csv')
X=card_data.iloc[:,0:7].values
y=card_data.iloc[:,7]

#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=786756)

################################################################################
#######
#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)

accuracy=accuracy_score(y_test, y_pred)

print("Linear Kernal Accuracy: ", accuracy)

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

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

accuracy=accuracy_score(y_test, y_pred)

print("Polynomial Kernal Accuracy: ", accuracy)

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

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

```python
accuracy=accuracy_score(y_test, y_pred)

print("Radial Kernal Accuracy: ", accuracy)

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

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

accuracy=accuracy_score(y_test, y_pred)

print("Sigmoid Kernal Accuracy: ", accuracy)
```

SAS Output

accuracy
0.950249

accuracy
0.910448

accuracy
0.925373

SAS Code

```sas
proc import out=sasuser.card_data
file="\\vdi-fileshare01\UEMprofiles\017365554\Desktop\STAT
574\Data\card_transdata.csv"
dbms=csv replace;
run;

libname poly5 "\\vdi-fileshare01\UEMprofiles\017365554\Desktop\SAS Enterprise
Minor Sets\SVpoly5\Workspaces\EMWS1\emsave";

data polynomial_kernel;
set poly5.em_save_test;
match=(fraud=lowcase(EM_CLASSIFICATION));
run;

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

```
libname sig5 "\\vdi-fileshare01\UEMprofiles\017365554\Desktop\SAS Enterprise
Minor Sets\SVsig5\Workspaces\EMWS1\emsave";

data sigmoid_kernel;
set sig5.em_save_test;
match=(fraud=lowcase(EM_CLASSIFICATION));
run;

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


libname rad5 "\\vdi-fileshare01\UEMprofiles\017365554\Desktop\SAS Enterprise
Minor Sets\SVrad5\Workspaces\EMWS1\emsave";

data radial_kernel;
set rad5.em_save_test;
match=(fraud=lowcase(EM_CLASSIFICATION));
run;

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

Problem 6. Use the data in the file “concussions_data.csv” to fit a support vector multinomial classifier. Specify linear, polynomial, radial, and sigmoid kernels (whichever are possible to fit). Compute and compare prediction accuracies. Compute prediction accuracy.

R Output

```
[1] "accuracy= 0.8879"
[1] "accuracy= 0.8276"
[1] "accuracy= 0.8879"
[1] "accuracy= 0.8879"
```

R Code

```
# Problem 6
# support vector multinomial classification
 # same kernals as above and compute prediction accuracy for each

concuss <- read.csv("C:/Users/saedw/OneDrive/Desktop/STAT 574 Data Mining/HW1STAT574S23/DATA SETS/concussions_data.csv")


#SPLITTING DATA INTO 80% TRAINING AND 20% TESTING SETS
```

```
set.seed(444625)
sample <- sample(c(TRUE, FALSE), nrow(concuss), replace=TRUE, prob=c(0.8,0.2))
train<- concuss[sample,]
test<- concuss[!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(concussion) ~ ., data=train, kernel="linear")

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

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


##################################################################
#FITTING SVM WITH POLYNOMIAL KERNEL
svm.multiclass<- svm(as.factor(concussion) ~ ., data=train, kernel="polynomial")

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

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


##################################################################
#FITTING SVM WITH RADIAL KERNEL
svm.multiclass<- svm(as.factor(concussion) ~ ., data=train, kernel="radial")

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

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

##################################################################
#FITTING SVM WITH SIGMOID KERNEL
svm.multiclass<- svm(as.factor(concussion) ~ ., data=train, kernel="sigmoid")
```

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

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

Python Output

```
Linear Kernal Accuracy:  0.8380952380952381
Polynomial Kernal Accuracy:  0.8857142857142857
Radial Kernal Accuracy:  0.8952380952380953
Sigmoid Kernal Accuracy:  0.44761904761904764
```

Python Output

```
# Problem 6 - support vector multinomial classifier
    # kernals: linear, polynomial, radial, sigmoid - compute accuracies
import pandas
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC

concussion_data=pandas.read_csv(r'C:\Users\saedw\OneDrive\Desktop\STAT 574 Data
Mining\HW1STAT574S23\DATA SETS\concussions_data.csv')

code_position={'Offensive Lineman': 1, 'Cornerback': 2, 'Running Back': 3,'Wide
Receiver': 4,
'Quarterback': 5}
code_concussion={'mild': 1, 'moderate': 2, 'severe': 3}

concussion_data['position']=concussion_data['position'].map(code_position)
concussion_data['concussion']=concussion_data['concussion'].map(code_concussion)


X=concussion_data.iloc[:,0:4]
y=concussion_data.iloc[:,4]

#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=599555)

#################################################################################
#######
#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)
```

```python
accuracy=accuracy_score(y_test, y_pred)

print("Linear Kernal Accuracy: ", accuracy)

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

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

accuracy=accuracy_score(y_test, y_pred)

print("Polynomial Kernal Accuracy: ", accuracy)

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

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

accuracy=accuracy_score(y_test, y_pred)

print("Radial Kernal Accuracy: ", accuracy)

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

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

accuracy=accuracy_score(y_test, y_pred)

print("Sigmoid Kernal Accuracy: ", accuracy)
```