République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université des Sciences et de la Technologie Houari Boumediene

Faculté de Mathématiques

Département de Probabilité/Statistiques

Project

Data Mining

Topic

Support vector Machine for a Multiclass classification task

Student : Maguedouda Samia BELHADDAD
Date : January 30 , 2023

Content

Project Description

Support Vector Machines is a Statistical Learning Technique that can be used for several tasks: Linear and Non-Linear Classification and Regression.

Which makes it a powerful Multi Usage Technique for modeling and prediction.

They are also suited for the classification of complex small or medium-sized datasets.

The objective is to build a multiclass classification model using Support Vector Machines.

The Data is taken from Kaggle.

https://www.kaggle.com/datasets/iabhishekofficial/mobile-price-classification?resource=download

MetaData:

id: ID

battery_power: Total energy a battery can store in one time measured in mAh

blue: Has Bluetooth or not

clock_speed: the speed at which microprocessor executes instructions

dual_sim: Has dual sim support or not

fc: Front Camera megapixels

four_g: Has 4G or not

int_memory: Internal Memory in Gigabytes

m_dep: Mobile Depth in cm

mobile_wt: Weight of mobile phone

n_cores: Number of cores of processor

pc: Primary Camera megapixels

px_height: Pixel Resolution Height

px_width: Pixel Resolution Width

ram: Random Access Memory in Megabytes

sc_h: Screen Height of mobile in cm

sc_w: Screen Width of mobile in cm

talk_time: longest time that a single battery charge will last when you are

three_g: Has 3G or not

touch_screen: Has touch screen or not

wifi: Has wifi or not

Section 1: Overview of Statistical Learning and Supervised Learning

1. Statistical Learning

Statistical learning refers to the process of using statistical methods to learn from data and make predictions or decisions.

It involves using mathematical models, such as linear regression, decision trees, and neural networks, to analyze data and extract insights.

The goal of statistical learning is to identify patterns in the data that can be used to make predictions or decisions about new, unseen data.

Statistical learning can be divided into two main categories: supervised learning and unsupervised learning.

Statistical learning is an important field in both statistics and machine learning, and it has many practical applications in fields such as finance, healthcare, marketing, and engineering.

2. Supervised Learning

Supervised learning involves using labeled data, where the outcome or target variable is known, to train a model that can predict the outcome for new, unseen data.

Examples of supervised learning techniques include linear regression and k-nearest neighbors.

3. Unsupervised Learning

Unsupervised learning, on the other hand, involves using unlabeled data, where the outcome or target variable is not known, to discover patterns or structures in the data.

Examples of unsupervised learning techniques include clustering and dimensionality reduction.

4. Multiclass Classification

Multiclass classification is a supervised learning problem where an algorithm is trained to predict one of multiple possible classes or labels for a given input data.

It is a generalization of binary classification, which only involves two classes.

For example, image classification tasks such as recognizing handwritten digits or objects in an image can be considered multiclass classification problems, as there are multiple possible digits or objects that the algorithm must be able to distinguish between.

A simple example of multiclass classification is classifying a set of observations into the values taken by a discrete categorical target variable.

Section 2: How Support Vector Machines work

1. Support Vector Machines Techniques

Support Vector Machines (SVMs) are a type of supervised learning algorithm that can be used for classification and regression tasks.

The Idea behind SVMs is to find the best boundary (also known as a hyperplane) that separates different classes in a dataset. The boundary is chosen in such a way that it maximizes the margin, which is the distance between the boundary and the closest data points from each class. SVMs are particularly useful when the data is not linearly separable, and kernel functions can be used to project the data into a higher dimensional space where a linear boundary can be found.

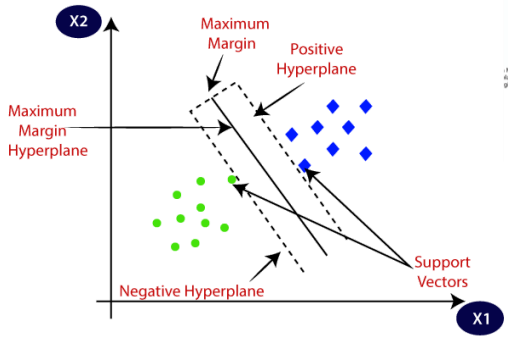


Figure 1: Support Vector Machine Illustration for Binary Classification

Support vector machines (SVMs) can be used for multiclass classification by using a technique called one-vs-all (also known as one-vs-rest).

This involves training a separate binary SVM classifier for each class, where the class is treated as the positive class and all other classes are combined into a single negative class. During prediction, all classifiers are run on the input and the class with the highest confidence score is chosen as the output. Another method that can be used for multiclass classification with SVMs is the multi-class reduction to binary (MC-SVM) method which constructs a unique SVM for each pair of classes and then combines these binary classifiers to form the final multiclass classifier.

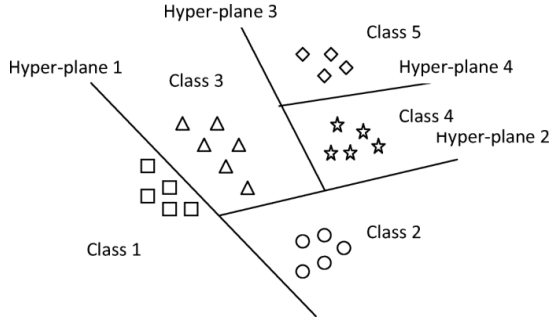


Figure 2: Support Vector Machine illustration for multiclass classification

Section 3: Data Preprocessing, Exploration, and preparation

About the Data

The Raw dataset is composed of one CSV file named Train data and taken from kaggle .
View of the file gives the following:

	battery_power	blue	clock_speed	dual_sim	fc	four_g	int_memory	m_dep	mobile_wt	n_cores	pc	px_height
1	842	0	2.2	0	1	0	7	0.6	188	2	2	20
2	1021	1	0.5	1	0	1	53	0.7	136	3	6	905
3	563	1	0.5	1	2	1	41	0.9	145	5	6	1263
4	615	1	2.5	0	0	0	10	0.8	131	6	9	1216
5	1821	1	1.2	0	13	1	44	0.6	141	2	14	1208
6	1859	0	0.5	1	3	0	22	0.7	164	1	7	1004
7	1821	0	1.7	0	4	1	10	0.8	139	8	10	381

px_width	ram	sc_h	sc_w	talk_time	three_g	touch_screen	wifi	price_range
756	2549	9	7	19	0	0	1	1
1988	2631	17	3	7	1	1	0	2
1716	2603	11	2	9	1	1	0	2
1786	2769	16	8	11	1	0	0	2
1212	1411	8	2	15	1	1	0	1
1654	1067	17	1	10	1	0	0	1
1018	3220	13	8	18	1	0	1	3

The dimensions of the raw dataset are

```
> dim(data)
[1] 2000   21
```

Training: We have 2000 Observations and 21 Variables

1. Problem and Solution

The problem here is to predict the price range of a specific mobile based on its features.

The solution is a Multiclass Classification.

The objective is to build a model that can predict the price of a mobile based on its feature.

The price range is categorical discrete with several classes which means it's a multiclass classification

2. Data Preprocessing

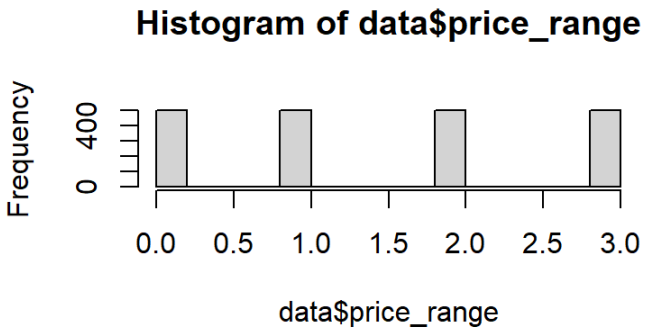
The target variable price_range is a variable with 4 possible classes which is the reason we chose multiclass classification.

```
> # View values of price_range
> unique(data$price_range)
[1] 1 2 3 0
```

There are no missing values in the train and test data.

```
> # Check for missing values
> sum(is.na(data))
[1] 0
```

3. Data Visualization



The histogram shows that the target variable is balanced in terms of the frequency of classes.

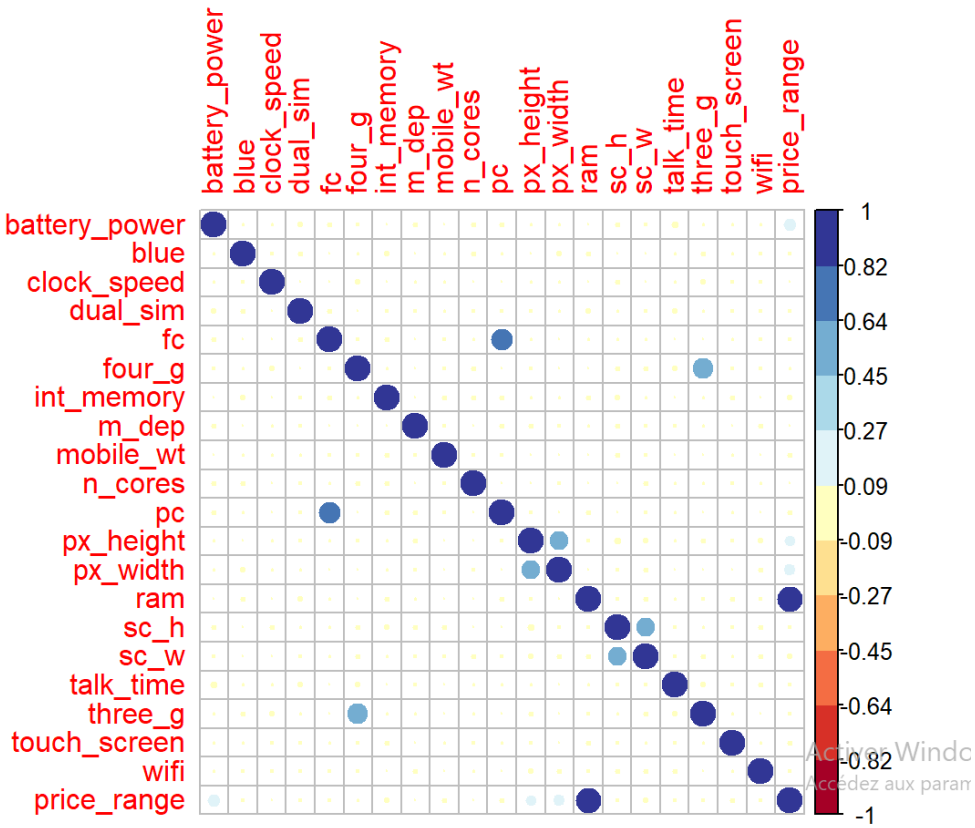
This means that the entire dataset is balanced in training.

which is a good thing as imbalanced classes cause a bias towards the class with the highest frequency in every classification task as it comes with the highest probability which is probabilistically logical.

4. Exploratory Data Analysis

Correlation Analysis

The results are shown in the following correlation heatmap:



From the Matrix, we can observe that:

Target Variable: Price range is highly correlated with:

Ram, Battery Power, px_height, px_width .

Feature Selection

Since the Price Range is highly correlated with the following: features

Ram, Battery Power,px_height, and px_width

Based on market research done:

I observed that features like Internal Memory and n_cores are also relevant in the price range of the mobile phone.

So I could include them in my study.

we can use the feature space defined by these features:

Features or Predictors: Ram, Battery Power, px_height and px_width,n_cores,int_memory

Target variable: price_range

The rest of the features will be dropped.

```
# new dataframe
data <- data[,c('battery_power','ram','int_memory','n_cores','px_height','px_width','price_range')]
View(data)
```

The new data frame is

	battery_power	ram	int_memory	n_cores	px_height	px_width	price_range
1	842	2549	7	2	20	756	1
2	1021	2631	53	3	905	1988	2
3	563	2603	41	5	1263	1716	2
4	615	2769	10	6	1216	1786	2
5	1821	1411	44	2	1208	1212	1
6	1859	1067	22	1	1004	1654	1
7	1821	3220	10	8	381	1018	3

Descriptive Statistics for Feature Space

```
> # Descriptive Statistics
> library(summarytools)
> descr(data, headings = FALSE, # remove headings
+       stats = "common" # most common descriptive statistics
+ )

                battery_power   int_memory   n_cores   price_range   px_height   px_width       ram
--------------- --------------- ------------ --------- ------------- ----------- ---------- ---------
           Mean         1238.52        32.05      4.52          1.50      645.11    1251.52   2124.21
        Std.Dev          439.42        18.15      2.29          1.12      443.78     432.20   1084.73
            Min          501.00         2.00      1.00          0.00        0.00     500.00    256.00
         Median         1226.00        32.00      4.00          1.50      564.00    1247.00   2146.50
            Max         1998.00        64.00      8.00          3.00     1960.00    1998.00   3998.00
        N.Valid         2000.00      2000.00   2000.00       2000.00     2000.00    2000.00   2000.00
      Pct.Valid          100.00       100.00    100.00        100.00      100.00     100.00    100.00
> |
```

5. Data Preparation

Train Test Split:

The advised ratio of train test splitting is always 80-20% of data

Feature Scaling:

Because Support Vector Machine (SVM) optimization occurs by minimizing the decision vector w, the optimal hyperplane is influenced by the scale of the input features and it's therefore recommended that data be standardized (mean 0, var 1) prior to SVM model training.

We manually scale using Caret Package

```
# Normalize train data between 0 and 1
# normalizing data
library(caret)
xtrain <- preProcess(x_train, method=c("range"))

x_train_scaled<- predict(xtrain,x_train )


x_test_sclaed <- predict(xtrain, x_test)
# summary stats after scaling
descr(X_train_scaled, headings = FALSE, # remove headings
      stats = "common" # most common descriptive statistics
)
descr(X_test_scaled, headings = FALSE, # remove headings
      stats = "common" # most common descriptive statistics
)
```

Preview of Data after Scaling :

x_test	408 obs. of 6 variables
x_test_sclaed	408 obs. of 6 variables
$ battery_power: num	0.0762 0.9071 0.8818 0.0628 0.181 ...
$ ram : num	0.672 0.217 0.792 0.934 0.685 ...
$ int_memory : num	0.129 0.323 0.129 0.339 0.597 ...
$ n_cores : num	0.714 0 1 0.286 0.857 ...
$ px_height : num	0.62 0.512 0.194 0.225 0.67 ...
$ px_width : num	0.858 0.77 0.346 0.207 0.904 ...
x_train	1592 obs. of 6 variables
x_train_scaled	1592 obs. of 6 variables
$ battery_power: num	0.2278 0.3474 0.0414 0.8818 0.9706 ...
$ ram : num	0.613 0.635 0.627 0.309 0.119 ...
$ int_memory : num	0.0806 0.8226 0.629 0.6774 0.3548 ...
$ n_cores : num	0.143 0.286 0.571 0.143 0.429 ...
$ px_height : num	0.0102 0.4617 0.6444 0.6163 0.2612 ...
$ px_width : num	0.171 0.993 0.812 0.475 0.433 ...

Summary Statistics after Scaling :

```
                battery_power   int_memory   n_cores   px_height   px_width       ram
--------------- -------------- ------------ --------- ----------- ---------- ---------
           Mean           0.49         0.49      0.50        0.33       0.50      0.50
        Std.Dev           0.29         0.29      0.32        0.22       0.29      0.29
            Min           0.00         0.00      0.00        0.00       0.00      0.00
         Median           0.48         0.48      0.43        0.29       0.50      0.50
            Max           1.00         1.00      1.00        1.00       1.00      1.00
        N.Valid        1592.00      1592.00   1592.00     1592.00    1592.00   1592.00
      Pct.Valid         100.00       100.00    100.00      100.00     100.00    100.00
> descr(x_test_scaled, headings = FALSE, # remove headings
+       stats = "common" # most common descriptive statistics
+ )

                battery_power   int_memory   n_cores   px_height   px_width       ram
--------------- -------------- ------------ --------- ----------- ---------- --------
           Mean           0.51         0.48      0.51        0.33       0.52     0.49
        Std.Dev           0.30         0.29      0.34        0.23       0.28     0.29
            Min           0.00         0.00      0.00        0.00       0.00     0.00
         Median           0.50         0.47      0.57        0.29       0.49     0.51
            Max           1.00         1.00      1.00        0.99       1.00     1.00
        N.Valid         408.00       408.00    408.00      408.00     408.00   408.00
      Pct.Valid         100.00       100.00    100.00      100.00     100.00   100.00
```

Prepare final input to models:

```
# Get features and Target Variable

x_dat = data.frame(x_train_scaled, y = as.factor(y_train))

x_dat_test <-data.frame(x_test_scaled, y_test)
```

Section 4: Modeling with SVM :

a. Support Vector Machine: Simple Training and Evaluation

In the call to svm() we'll use the formula price_range~. which indicates we want to classify the price_range response variable using the 6 other predictors found in the data set.

We also specify the type of usag for svm() with type="C-classification" .

We use kernel="radial" (the default) for this multi-class classification problem.

We can tune the operation of svm() with two additional arguments: gamma and cost, where gamma is the argument for use by the kernel function, and cost allows us to specify the cost of a violation to the margin.

```
#SVM Classifier
library(e1071)

svm <- svm(formula = y~., data=x_dat,
              method="C-classification", kernal="radial",
              gamma=0.1, cost=10)

summary(svm)
svm$SV
svm$labels
```

Summary of SVM Classifier :

```
> summary(svm)

Call:
svm(formula = y ~ ., data = x_dat, method = "C-classification", kernal = "radial",
    gamma = 0.1, cost = 10)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  10

Number of Support Vectors:  443

 ( 148 149 69 77 )


Number of Classes:  4

Levels:
 0 1 2 3
```

```
> svm$SV[0:10]
 [1] -0.8928110  1.3466890 -1.2588171 -1.6682865
 [5]  0.1983448 -1.1604530  0.9692452  0.6444148
 [9]  1.2002866 -0.5130796
> svm$labels
[1] 2 3 1 4
```

Training and test Accuracy :

```
# compute training accuracy
pred_train <- predict(svm, x_dat)
mean(pred_train == x_dat$y)


#compute test accuracy
pred_test <- predict(svm,x_dat_test)
mean(pred_test == x_dat_test$y_test)
```

Training Accuracy is 0.9780

Testing Accuracy is 0.9240

```
> #Model Evaluation
> # compute training accuracy
> pred_train <- predict(svm, x_dat)
> mean(pred_train == x_dat$y)
[1] 0.9780151
> # compute test accuracy
> pred_test <- predict(svm,x_dat_test)
> mean(pred_test == x_dat_test$y_test)
[1] 0.9240196
>
```

We can observe that the accuracy is :
0.978 for training.
0.924 for testing .

Confusion matrix

```
> conf_mat <- confusionMatrix(table(x_dat_test$y_test,
+                                   predictions = pred_test))
> conf_mat
```

```
> conf_mat
Confusion Matrix and Statistics

   predictions
     0  1  2  3
  0 96  4  0  0
  1  3 91  4  0
  2  0  9 98  4
  3  0  0  7 92

Overall Statistics

               Accuracy : 0.924
                 95% CI : (0.8939, 0.9478)
    No Information Rate : 0.2672
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8986
```

Classification Report

```
Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3
Sensitivity            0.9697   0.8750   0.8991   0.9583
Specificity            0.9871   0.9770   0.9565   0.9776
Pos Pred Value         0.9600   0.9286   0.8829   0.9293
Neg Pred Value         0.9903   0.9581   0.9630   0.9871
Prevalence             0.2426   0.2549   0.2672   0.2353
Detection Rate         0.2353   0.2230   0.2402   0.2255
Detection Prevalence   0.2451   0.2402   0.2721   0.2426
Balanced Accuracy      0.9784   0.9260   0.9278   0.9679
```

The model is able to predict classes 2 and 1 really well.

Classification Report :

Sensitivity: true positive rate: it measures the proportion of actual positive cases that are correctly identified as such.

Specificity: true negative rate, it measures the proportion of actual negative cases that are correctly identified as such.

Positive Predictive Value (PPV): also known as precision, it measures the proportion of positive cases identified by a test that are actually positive.

Negative Predictive Value (NPV): measures the proportion of negative cases identified by a test that are actually negative.

Prevalence: measures the proportion of the population that has a specific condition.

Detection Rate: also known as hit rate, it measures the proportion of actual positive cases that are identified as such.

Detection Prevalence: measures the proportion of the population that is identified as having a specific condition.

Balanced Accuracy: it is the average of sensitivity and specificity, used to measure a model's overall accuracy by taking into account the imbalance of classes in dataset

b. Support Vector Machine Model: GridSearch Cross Validation

Grid search cross-validation is a technique used to tune the hyperparameters of a machine learning model. It involves specifying a set of possible values for each hyperparameter, and then training and evaluating the model using all possible combinations of these values.

This process is typically done using k-fold cross-validation, where the data is split into k subsets and the model is trained and evaluated k times, each time using a different subset as the validation set.

The combination of hyperparameters that results in the best performance on the validation set is then chosen as the final model.

Model 1 Trial: Sigma = c(0.1, 1, 10), C = c(0.1, 1, 10) :

```
# Define the parameter grid
param_grid <- expand.grid(sigma = c(0.1, 1, 10), C = c(0.1, 1, 10))
x_dat[,1:6]
# Perform the grid search with 5-fold cross-validation
svm_grid <- train(x = x_dat[,1:6], y = x_dat$y, method = "svmRadial",
                  tuneGrid = param_grid, trControl = trainControl(method = "cv", number = 5))

# Print the results
print(svm_grid)
```

Results :

```
1592 samples
   6 predictor
   4 classes: '0', '1', '2', '3'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 1274, 1273, 1274, 1273, 1274
Resampling results across tuning parameters:

  sigma  C     Accuracy   Kappa
   0.1    0.1  0.8963487  0.861808182
   0.1    1.0  0.9265137  0.902005227
   0.1   10.0  0.9221151  0.896140741
   1.0    0.1  0.6576369  0.542270779
   1.0    1.0  0.8454782  0.793958177
   1.0   10.0  0.8442105  0.792265523
  10.0    0.1  0.2543976  0.003361464
  10.0    1.0  0.4974784  0.328397772
  10.0   10.0  0.5207173  0.359552884

Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.1 and C = 1.
```

The best Parameters are sigma=0.1 and C=1

We use them to train the SVM Classifier .

SVM classifier with sigma = 0.1 and C= 1 :

```
> # Define the model (SVM with radial kernel)
> svm <- svm(formula = y~., data=x_dat,
+            method="C-classification", kernal="radial",
+            gamma=0.1, cost=1)
> #Model Evaluation
> # compute training accuracy
> pred_train <- predict(svm, x_dat)
> mean(pred_train == x_dat$y)
[1] 0.9610553
> # compute test accuracy
> pred_test <- predict(svm,x_dat_test)
> mean(pred_test == x_dat_test$y_test)
[1] 0.9289216
```

Training and Test Accuracy are 0.96 and 0.9289

We can observe that the accuracy of test is better which is better .

Because we want the training accuracy to be good enough but the test accuracy to be high as the model should be able to generalize to new data (test).

Confusion Matrix:

```
> conf_mat <- confusionMatrix(table(x_dat_test$y_test,
+                                   predictions = pred_test))
> conf_mat
Confusion Matrix and Statistics

   predictions
     0  1  2  3
  0 97  3  0  0
  1  5 90  3  0
  2  0  8 97  6
  3  0  0  4 95

Overall Statistics

               Accuracy : 0.9289
                 95% CI : (0.8995, 0.9519)
    No Information Rate : 0.2549
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9052
```

The model is able to predict class 0 and 2 better than the first model

Classification Report

```
Statistics by Class:

                     Class: 0 Class: 1 Class: 2 Class: 3
Sensitivity            0.9510   0.8911   0.9327   0.9406
Specificity            0.9902   0.9739   0.9539   0.9870
Pos Pred Value         0.9700   0.9184   0.8739   0.9596
Neg Pred Value         0.9838   0.9645   0.9764   0.9806
Prevalence             0.2500   0.2475   0.2549   0.2475
Detection Rate         0.2377   0.2206   0.2377   0.2328
Detection Prevalence   0.2451   0.2402   0.2721   0.2426
Balanced Accuracy      0.9706   0.9325   0.9433   0.9638
>
```

Model 2 Trial: sigma = c(0.1, 1,7, 12,10), C = c(0.1, 1,5,0.6,15,20, 2)

Grid Search cross validation gave the following

```
# Grid 2 :
# Define the parameter grid
param_grid <- expand.grid(C = c(0.1, 1,5,0.6,15,20, 2),sigma = c(0.1, 1,7, 12,10))
```

```
> # Perform the grid search with 5-fold cross-validation
> svm_grid <- train(x= x_dat[,1:6], y = x_dat$y, method = "svmRadial", tuneGrid = param_grid, trControl = trainC
ntrol(method = "cv", number = 5))
> print(svm_grid)
Support Vector Machines with Radial Basis Function Kernel

1592 samples
   6 predictor
   4 classes: '0', '1', '2', '3'

No pre-processing
Resampling: Cross-Validated (5 fold)
Summary of sample sizes: 1273, 1274, 1273, 1273, 1275
Resampling results across tuning parameters:

  C     sigma  Accuracy   Kappa
   0.1   0.1   0.9032250  0.8709755191
   0.1   1.0   0.6532666  0.5364840804
   0.1   7.0   0.2600353  0.0109032643
   0.1  10.0   0.2525118  0.0008429593
   0.1  12.0   0.2518848  0.0000000000
   0.6   0.1   0.9253066  0.9137354319
```


The highest accuracy is found for sigma =0.1 and C=15 :

```
Accuracy was used to select the optimal model using the largest value.
The final values used for the model were sigma = 0.1 and C = 15.
```

Summary of the SVM model :

```
Call:
svm(formula = y ~ ., data = x_dat, method = "C-classification", kernal = "radial", gamma = 0.1,
    cost = 15)


Parameters:
   SVM-Type:  C-classification
 SVM-Kernel:  radial
       cost:  15

Number of Support Vectors:  406

 ( 132 138 64 72 )


Number of Classes:  4

Levels:
 0 1 2 3
```


Model Evaluation :

```
> #Model Evaluation
> # compute training accuracy
> pred_train <- predict(svm, x_dat)
> mean(pred_train == x_dat$y)
[1] 0.9836683
> # compute test accuracy
> pred_test <- predict(svm,x_dat_test)
> mean(pred_test == x_dat_test$y_test)
[1] 0.9215686
```

We can observe that the training accuracy is higher than the first two models but the test accuracy is lower .
This means that this model is better at learning patterns in the training data and is relatively not better than the first two models for generalizing to new data .

Confusion Matrix :

```
Confusion Matrix and Statistics

   predictions
     0  1  2  3
  0 96  4  0  0
  1  2 92  4  0
  2  0 10 96  5
  3  0  0  7 92

Overall Statistics

               Accuracy : 0.9216
                 95% CI : (0.8911, 0.9457)
    No Information Rate : 0.2623
    P-Value [Acc > NIR] : < 2.2e-16
```

Final Results

Support vector Machines are a good tool for Multiclass Classification of Tabular Data.

Support vector machines give a high accuracy for mobile price prediction.

Grid Search Cross Validation helps get the best parameters for the Classifier.

By comparing 3 models, we can see that the second model was the best classifier.