Q1:

(a)Vehicle Silhouette Dataset:

The data set consists of 17 numeric attributes each pertaining to a feature extracted from the vehicle silhouette. We, then, use this set of attributes to classify a given silhouette as one of four types of vehicle: OPEL, BUS, SAAB, and VAN.

We convert the supervised learning problem into active learning problem by randomly removing class labels of data points and retaining only 10% of labelled data points as follows:

```
library(readxl)

vehicle_dataset <- read_excel("vehicle_dataset.xlsx")

str(vehicle_dataset)

vehicle_dataset$Class<-as.factor(vehicle_dataset$Class)

vehicle_dataset[,1:18]<-scale(vehicle_dataset[,1:18],TRUE,TRUE)



library(dplyr)

library(caTools)

set.seed(1234)

s1 <-sample.split(vehicle_dataset$Class,SplitRatio = 0.90)

t1<-cbind(vehicle_dataset,s=s1)

unlabelledData<-subset(t1,s==TRUE)[1:19]

labelledData<-subset(t1,s==FALSE)[1:19]

unlabelledData1<-unlabelledData

labelledData1<-labelledData
```

We scale and pre-process the data using linear discriminant analysis from the previous assignment.

```
library(MASS)

model<-lda(Class~.,data=labelledData)

labelledData.lda<-predict(model,newdata =as.data.frame(labelledData) )$x

unlabelledData.lda<-predict(model,newdata = as.data.frame(unlabelledData[,1:18]))$x

labelledData.lda<-cbind.data.frame(labelledData.lda,"Class"=labelledData$Class)

unlabelledData.lda<-cbind.data.frame(unlabelledData.lda,"Class"=unlabelledData$Class)
```

(b)

i) Uncertainty Sampling:

Least Confident:

Naïve-Bayes classifier is used for calculation of the class probabilities.
In the algorithm below, we first calculate class-wise probabilities of each row.
Then we find the maximum class probability in each row. The data is then arranged in an increasing order and the row which is the most difficult to classify i.e. the row with minimum value of maximum class probability is labelled and added to the labelled dataset.

```
library(naivebayes)
##Least Confidence
for(i in c(1:(nrow(vehicle_dataset)*percent))){
        nb<-naive_bayes(Class~.,labelledData.lda)
        xtab<-table(predict(nb,unlabelledData.lda, type="class"),unlabelledData.lda$Class)
        acc=sum(diag(xtab))/nrow(unlabelledData.lda)*100
        print(acc)  a<-predict(nb,unlabelledData.lda, type="prob")
        unlabelledData.lda<-cbind(unlabelledData.lda,"max"=apply(a, 1, max))
        unlabelledData.lda<-unlabelledData.lda %>% arrange(max)
        labelledData.lda<-rbind(labelledData.lda,unlabelledData.lda[1,1:4])
        unlabelledData.lda<-unlabelledData.lda[2:nrow(unlabelledData.lda),1:4]
}
```

Initial accuracy of the Naïve-Bayes classifier is 72.14%

Additional points labelled (percent)	**Accuracy**
10%	76.7%
20%	79.76%
30%	84.09%
40%	90.33%

Margin Sampling

For margin sampling, after building a Naïve-Bayes classifier we find class probabilities and find the difference of the two maximum class probabilities for each data entry. We, then, select the data entry with the least difference between the two maximum class probabilities.

```
#MarginSampling
for(i in c(1:(nrow(vehicle_dataset)*percent))){
        nb<-naive_bayes(Class~.,labelledData.lda)
        xtab<-table(predict(nb,unlabelledData.lda, type="class"),unlabelledData.lda$Class)
        acc=sum(diag(xtab))/nrow(unlabelledData.lda)*100
        print(acc)  a1<-predict(nb,unlabelledData.lda, type="prob")
        maxes<-t(sapply(1:nrow(a1),function(i){
                sort(a1[i,1:4],decreasing = TRUE)[1:2]
```

```
        }))
        diff<-maxes[,1]-maxes[,2]
        unlabelledData.lda<-cbind(unlabelledData.lda,"diff"=diff)
        unlabelledData.lda<-unlabelledData1.lda %>% arrange(diff)
        labelledData1.lda<-rbind(labelledData1.lda,unlabelledData1.lda[1,1:4])
        unlabelledData1.lda<-unlabelledData1.lda[2:nrow(unlabelledData1.lda),1:4]
}
```

Initial accuracy of the Naïve-Bayes classifier is 72.14%

Additional points labelled (percent)	**Accuracy**
10%	76.99%
20%	80.78%
30%	84.68%
40%	90.35%

Entropy Sampling

Entropy sampling takes into account all the class probabilities. Entropy is calculated as :

$$entropy = -\sum_{i=1}^{n} p_i log_2(p_i)$$

After calculating entropy row-wise, all the rows of data are sorted according to entropy values and the row with the highest entropy value is labelled and added to the labelled training set.

```
#Entropy Sampling
for(i in c(1:(nrow(vehicle_dataset)*percent))){
        nb<-naive_bayes(Class~.,labelledData.lda)
        xtab<-table(predict(nb,unlabelledData.lda, type="class"),unlabelledData.lda$Class)
        acc=sum(diag(xtab))/nrow(unlabelledData.lda)*100
        print(acc)  a<-predict(nb,unlabelledData.lda, type="prob")
        a.log<-log2(a)
        unlabelledData.lda<-cbind(unlabelledData.lda,"entropy"=rowSums(-a.log * a))
        unlabelledData.lda<-unlabelledData.lda %>% arrange(entropy)
        labelledData.lda<-rbind(labelledData.lda,unlabelledData.lda[nrow(unlabelledData.lda),1:4])
        unlabelledData.lda<-unlabelledData.lda[1:(nrow(unlabelledData.lda)-1),1:4]
}
```

Initial accuracy of the Naïve-Bayes classifier is 72.14%

Additional points labelled (percent)	**Accuracy**
10%	75.84%
20%	78.84%

30%	82.35%
40%	87.74%

Comparison of three measures of informativeness:

Out of the three factors, margin sampling seems to be the most informative. Although one might feel theoretically that entropy sampling would be the most suitable however, it is not so. This might be due to overfitting of the data by entropy sampling. Entropy sampling turns out to be the least accurate of the three.

V.) Clustering using K-means

From the 90% of the unlabelled points we first select 40% of points to apply clustering

```
train_set<-unlabelledData[sample(nrow(unlabelledData),nrow(unlabelledData)*0.4),]

train_set.lda<-predict(model,newdata = as.data.frame(train_set[,2:19]))$x

train_set.lda<-cbind(train_set.lda,"id"=train_set[,1])
```

Clustering using K-means algorithm. Number of classes, k=4.

Algorithm

I. First, we select 4 random points as cluster centroids.
II. We calculate the distance of each point/ row from these 4 possible centroids and classify the point to a cluster with which it is most close to.
Distance is calculated as Euclidean distance.

$$\text{Distance}=\sqrt{(p_1-q_1)^2+(p_2-q_2)^2+(p_3-q_3)^2 \dots \dots \dots}.$$

Sum of all the distances of all the points from the cluster centroids is kept as a measure of error.
III. Then a new centroid is found by taking the mean of all the points in each cluster.
IV. The process is repeated until the sum of all the distances from the cluster centroids does not change which in this case is don't for 20 iterations.

```
k<-4

#randomly selecting k points as centroids.
c<-train_set.lda[sample(nrow(train_set.lda),k),]

for(j in c(1:20)){
        train_set.lda<-train_set.lda[,1:4]
        Dc<-c(rowSums(sweep(train_set.lda[,1:3],2,c[1,1:3])*sweep(train_set.lda[,1:3],2,c[1,1:3])))
        cl<-c(rep(1,nrow(train_set.lda)))
        train_set.lda<-cbind.data.frame(train_set.lda,Dc=Dc)
        train_set.lda<-cbind.data.frame(train_set.lda,cl)
        for(i in c(2:4)){
                Dc<-rowSums(sweep(train_set.lda[,1:3],2,c[i,1:3])*sweep(train_set.lda[,1:3],2,c[i,1:3]))
                train_set.lda$cl<-if_else(train_set.lda$Dc>Dc,i,as.integer(train_set.lda$cl))
                train_set.lda$Dc<-if_else(train_set.lda$Dc>Dc,Dc,train_set.lda$Dc)
```

```
        }
         #now we take means of all the points in each cluster as new centroid
         for(i in c(1:4)){
                 c[i,]<-colMeans(filter(as.data.frame(train_set.lda)[,-5],cl==i)[,-5])
         }
         print(sum(train_set.lda$Dc))
}
```

After clustering of the data, we then randomly label 20% of the points in each cluster and assign the whole cluster to the class of the majority labels.

```
vehicle_dataset<-add_rownames(vehicle_dataset,"id")

#for first cluster and similarly for other clusters as well.
cluster1<-filter(train_set.lda[,-5],cl==1)[,-5]
cluster1.20<-cluster1[sample(nrow(cluster1),nrow(cluster1)*0.2),]
cluster1.20.labels<-filter(vehicle_dataset,as.numeric(vehicle_dataset$id) %in% cluster1.20$id)$Class
n1<-names(which.max(summary(cluster1.20.labels)))
cluster1.labels.actual<-filter(vehicle_dataset,as.numeric(vehicle_dataset$id) %in% cluster1$id)$Class
print(summary(cluster1.labels.actual)[n1])
```

The accuracy of the clustering algorithm is 76.64% as we are able to correctly classify 233 points out of 304 points in total.

Total number of points to be labelled = 304.
Cost of labelling each point = Rs. 100 + 1 hour of time.
Cost of labelling without clustering= Rs. 30400 +304 hours of time.

By using cluster-based labelling of 20% of the points,
We label 60 points in total.
Cost with clustering= Rs. 6000 + 60 hours of time

Therefore using cluster-based labelling results in a saving of Rs. 24400 and 244 hours of time. Thus, we infer that clustering can make the process of classification economically viable and less time-consuming with little compromise on the accuracy.