**Appendix**

**I. Distinction between SVM and SVM Perf**

The two statistical classifiers that we use are SVM and SVM Perf. SVM Perf is an implementation of the support vector machine (SVM). SVM is trained using error rate as the optimisation function, i.e., the goal of the classifier is to obtain error rate as low as possible. This may not be desirable in all situations. SVM Perf allows the optimisation function to be another metric (like precision, recall, F1-score, ROC area, etc.). In our case, we use SVM Perf optimised for F1-score (i.e., the goal of the classifier is to obtain F1-score as high as possible). F1-score provides a better indication than error rate because it accounts for precision (Accuracy) and recall (Coverage).

**II. Non-uniform mis-classification costs**

We use non-uniform mis-classification costs when learning the model. The mis-classification cost assigns a numeric cost for mis-classifying an instance. If the positive mis-classification cost is 2, the classifier is taught that mis-classifying a positive instance contributes to the optimisation metric (such as error) twice as much as mis-classifying a negative instance. This choice is because the positive/negative distribution is not uniform. ‘Positive’ here refers to tweets that are personal health mentions while ‘negative’ refers to the ones that are not.

**III. EWMA statistics**

EWMA stands for exponentially weighted moving average. The EWMA is a moving average that gives more recent observations greater weight in determining the average. The thresholds are determined using simulations assuming that the TBE are Weibull distributed. The process of finding these thresholds are outlined in Sparks et al (2019). The thresholds are designed to deliver a particular low false discovery rate. For the given parameters of the Weibull, threshold of 1 for the EWMA statistic indicates an alert.

**IV. Significance of consecutive alerts with false discovery rates**

Training the appropriate false discovery rate is challenging as can be seen in Sparks et al (2019). So, if there are usually too many events within a day for using Sparks et al (2019) results, then we insist on several consecutive signals as the way of controlling the false discovery rate to an acceptably low value. If no outbreak has occurred, it is more unlikely that several consecutive flagged events will occur. This is a much easier approach to controlling the false discovery rate rather than using a designed threshold for the that level of time between event frequency.

**V. Choice of Weibull distribution**

The distribution that are used in the literature are exponential, gamma and Weibull. The exponential is the simplest version. However, as shown in Sparks et al (2019), we observe that the Weibull distribution is the most appropriate in our example. We have shown the quartile-quartile plots for two keywords in eFigure 1. The points along the quarter line show that the Weibull distribution fits well. Since TBE is expected to be as low as possible for an outbreak to be detected, it also helps to see points towards the left bottom of the line to be lower than the reference diagonal shown in red.

|  |  |
| --- | --- |
|  |  |
| 1. ‘breath’ for Stat-SVM Perf | 1. ‘cough’ for Stat-SVM Perf |
| eFigure1: Sample quartile-quartile plots for two keywords. | |

**VI.** **Performance of StatSVM and StatSVMPerf**

In this section, we compare tweet vectors with vectors based on n-grams (specifically, unigrams and bigrams). We do so in two parts: (a) five-fold cross-validation, and (b) evaluation on the monitoring task.

We first obtain five-fold cross-validation performance for the two statistical classifiers on the dataset by Lamb et al (2013). This is the same dataset on which the personal health mention classification module is eventually trained as a part of the four-step architecture. eTable 1 shows the precision, recall F1-score values for the classifiers. The classifiers with word embeddings as the feature representation have a significantly higher recall as compared to n-grams as feature representation. The F-score is comparable (80 versus 77) in the case of StatSVM. A high recall indicates high coverage of positive tweets. Since it is critical for us to capture these personal health mentions, we prefer the feature representation that results in a high recall as the central contribution of the paper. Because n-gram-based representations rely on presence of words, these n-grams need to also be present in the test sentences for the classifiers to function well. Therefore, such classifiers may not generalise well. On the other hand, word embeddings provide a real-valued similarity between words (and resultantly, sentences) and do not require an exact match, since they are dense, real-valued vectors. However, it must be noted that this section of the Appendix reports results on all configurations.

eTable1: Performance of the statistical classifiers in percentage.

We now evaluate the four-step architecture for the n-gram-based representations in eTable 2. This means that, in the case of the Step 2, we use a statistical classifier using n-grams as features, while the other steps remain unchanged. We observe that the alerts are mostly after the known benchmark when n-grams are used as features. This shows that embedding-based classification results in a better performance as compared to n-gram-based classification.

eTable 2: Relevant Alerts Generated Using Different Dataset and Classifier Combinations; Alerts Within 5 Days of the Actual Acute Disease Event Mentioned. These results are for n-grams as features, as opposed to word embedding-based features in the main paper.

Similar to the main paper, we observe that ‘Other’ leads to alerts after the known date. The closest alert is obtained with ‘breath’, ‘SVM’ and ‘1 in 1000’ combination. However, this is not as good as the embedding-based classification that is reported in the main body of the paper. This highlights that embedding-based representations are a better option. In the related work section of the paper, we also point to past work that uses embedding-based sentence representations with a statistical classifier.

**VII. GloVe**

The GloVe algorithm takes word co-occurrence statistics as input and computes vectors for words. When trained on a large corpus, the GloVe embeddings capture meanings of words (Pennington et al., 2014). This means that vectors of words with similar meaning can be expected to have high cosine similarity.

**VIII. R Code of the Monitoring Algorithm**

The following is the code for the false discovery rate of 1 in 1000. For 1 in 2000, the newbeta equations (at two places) are changed to:

newbeta = 11.34264035-26.478708781\*al-15.600594242\*SHAPE[1]+5.318232968\*log(al)+ 7.508902272\*log(SHAPE[1])+4.226762238\*al^2-10.069510271\*al^3+4.954951662\*SHAPE[1]^2+0.002174711\*SHAPE[1]^3-0.330587359\* SHAPE[1]^4+35.873622054\*al\*SHAPE[1]-15.827524627\*al\*log(SHAPE[1])- 14.601662271\*al\*SHAPE[1]^2+3.081527759\*al\*SHAPE[1]^3-8.266950362\*SHAPE[1]\*log(al)+3.435500912\*log(al)\*log(SHAPE[1])+3.534716421\*log(al)\*SHAPE[1]^2-0.693932919\*log(al)\*SHAPE[1]^3-5.472227177\*al^2\*SHAPE[1]^2+7.514540350\*SHAPE[1]\*al^2 +1.968954215\*al^3\*SHAPE[1]^3

*Code begins here:*

data<-read.table(“<filename>",sep="$") # Insert filename here.

data[1:4,]

head(data)

data<-as.data.frame(data)

head(data)

data$V1<-as.Date(data$V1,format="%d.%m.%Y")

head(data)

data$yr <- as.numeric(as.vector(substring(data$V1,1,4)))

hr<-as.numeric(as.vector(substring(data$V2,1,2)))

min<-as.numeric(as.vector(substring(data$V2,4,5)))

sec<-as.numeric(as.vector(substring(data$V2,7,8)))

dtime<-hr+min/60+sec/3600

dtime[1:4]

data<-cbind(data,dtime)

wd<-weekdays(data$V1)

wd[1:4]

time<-as.numeric(as.vector(julian(data$V1)))

data<-cbind(data,time,wd)

min(data$time)

min(data$time)

summary(time)

data[is.na(data$time),]

data<-data[!is.na(data$time),]

mindatetime<-min(data$time)

data$time<-data$time-mindatetime

data$time<-data$time+dtime/24

data$time<-data$time+data$dtime/24

order(data$time)[1:1000]

dim(data)

sort(data$time)

ind<-order(data$time)

data<-data[ind,]

data$time

size <-length(data$time)

TBE<-data$time[-1]-data$time[-1\*size]

summary(TBE)

data$time[-1]==data$time[-1\*size]

TBE<-cbind(TBE,data[-1\*size,])

temp<-TBE[TBE$TBE == 0,]

TBE = TBE[!TBE$TBE == 0,]

library(gamlss)

fm<-gamlss(TBE~time+as.factor(yr)\*(cos(2\*pi\*time/365.25)+sin(2\*pi\*time/365.25))+wd\*(dtime+cos(2\*pi\*dtime/24)+sin(2\*pi\*dtime/24)+cos(2\*pi\*dtime/12)+sin(2\*pi\*dtime/12)), sigma.fo=~time+as.factor(yr)\*(cos(2\*pi\*time/365.25)+sin(2\*pi\*time/365.25))+wd\*(dtime+cos(2\*pi\*dtime/24)+sin(2\*pi\*dtime/24)+cos(2\*pi\*dtime/12)+sin(2\*pi\*dtime/12)),data=TBE,family=WEI())

sigma <- exp(predict(fm, what=c("sigma"), newdata=temp))

mu <- exp(predict(fm, what=c("mu"), newdata=temp))

q <- 1:dim(temp)[1]/dim(TBE)[1]

temp$TBE = qWEI(p=q,mu=mu,sigma=sigma)

t4 = rbind(TBE, temp)

t4<-t4[order(t4$time),]

fm<-gamlss(TBE~time+as.factor(yr)\*(cos(2\*pi\*time/365.25)+sin(2\*pi\*time/365.25))+wd\*(dtime+cos(2\*pi\*dtime/24)+sin(2\*pi\*dtime/24)+cos(2\*pi\*dtime/12)+sin(2\*pi\*dtime/12)), sigma.fo=~time+as.factor(yr)\*(cos(2\*pi\*time/365.25)+sin(2\*pi\*time/365.25))+wd\*(dtime+cos(2\*pi\*dtime/24)+sin(2\*pi\*dtime/24)+cos(2\*pi\*dtime/12)+sin(2\*pi\*dtime/12)),data=t4,family=WEI())

SCALE<-fm$mu.fv

SHAPE<-fm$sigma.fv

scale<-fm$mu.fv[(length(SCALE)+1):length(fm$mu.fv)]

shape<-fm$sigma.fv[(length(SCALE)+1):length(fm$mu.fv)]

par(mfrow=c(1,1))

SCALE<-c(SCALE,scale)

SHAPE<-c(SHAPE,shape)

TBE<-t4

h<-rep(0,length(SCALE))

x.mu<-SCALE\*gamma((1/SHAPE)+1)

ew<-x.mu

xmu<-SCALE[1]

al<-0.05

newbeta=15.4863461-31.2740388\*al-22.978514\*SHAPE[1]+6.1326295\*log(al)+ 9.3864181\*log(SHAPE[1])+ 8.1190440 \*al^2 -14.7686294\*al^3 +9.9260160\*SHAPE[1]^2 -1.9660931\*SHAPE[1]^3 +42.3619299\*al\*SHAPE[1]-17.5618988 \*al\*log(SHAPE[1]) -17.2181503 \*al\*SHAPE[1]^2+ 3.5450699\*al\*SHAPE[1]^3-9.5183478\*SHAPE[1]\*log(al)+3.8567301\*log(al)\*log(SHAPE[1])+4.0985140\*log(al)\*SHAPE[1]^2-0.8025912\*log(al)\*SHAPE[1]^3 -6.4936384\*al^2\*SHAPE[1]^2+6.4764153\* SHAPE[1]\*al^2+ 3.436274\*al^3\*SHAPE[1]^3

h[1]<-newbeta\*SCALE[1]

xmu<-min(0.1\*TBE$TBE[1]/gamma((1/SHAPE[1])+0.9\*xmu),SCALE[1])

ew[1]<-min(c(al\*TBE$TBE[1]/h[1]+(1-al)\*x.mu[1]/h[1],x.mu[1]/h[1]))

ew.last<-ew[1]

for(I in 2:length(SHAPE)-1){

alopt<-(-0.32699401+0.02981035\*xmu+0.27631529\*scale+0.03879301\*shape-0.17932719\*log(xmu)+0.0882661\*log(scale)+0.01238951\*xmu\*scale- 0.06672816\*xmu\*shape+0.17150974\* xmu\*log(xmu)+0.04734040\*scale\*shape+0.07158034\*scale\*log(xmu)-0.02679079\*shape\*log(xmu)-0.09427231\*scale\*log(scale)-0.29939506\*xmu\*log(scale)-0.01674911\* log(xmu)\*log(scale))

alopt[alopt<0.03]<-0.03

alopt[alopt>0.25]<-0.25

al<-max(c(0.02,alopt),na.rm=T)

al<-min(c(0.25,al),na.rm=T)

newbeta=15.4863461-31.2740388\*al-22.978514\*SHAPE[I]+6.1326295\*log(al)+ 9.3864181\*log(SHAPE[I])+ 8.1190440 \*al^2 -14.7686294\*al^3 +9.9260160\*SHAPE[I]^2 -1.9660931\*SHAPE[I]^3 +42.3619299\*al\*SHAPE[I]-17.5618988 \*al\*log(SHAPE[I]) -17.2181503 \*al\*SHAPE[I]^2+ 3.5450699\*al\*SHAPE[I]^3-9.5183478\*SHAPE[I]\*log(al)+3.8567301\*log(al)\*log(SHAPE[I])+4.0985140\*log(al)\*SHAPE[I]^2-0.8025912\*log(al)\*SHAPE[I]^3 -6.4936384\*al^2\*SHAPE[I]^2+6.4764153\* SHAPE[I]\*al^2+ 3.436274\*al^3\*SHAPE[I]^3

h[I]<-newbeta\*SCALE[I]

ew[I] <- min(c(al\*TBE$TBE[I]/h[I]+(1-al)\*ew.last,x.mu[I]/h[I]))

if(ew[I]<1)ew.last<-x.mu[I]/h[I] else ew.last<-ew[I-1]

xmu<-min(0.1\*TBE$TBE[I]/gamma((1/SHAPE[I])+1)+0.9\*xmu,SCALE[I])

}

plot(TBE$V1,ew[1:length(TBE$V1)],xlab="Date",ylab="Adaptive EWMA")

abline(h=1,col=2)

ew2<-ew<1

ewint <- ew2[1:length(TBE$V1)]

ewres <- numeric(length(TBE$V1))

for (i in 8:length(ewint)) {ewres[i] <- ewint[i-1] + ewint[i-2] + ewint[i-3] + ewint[i-4] + ewint[i-5] + ewint[i-6] + ewint[i-7]}

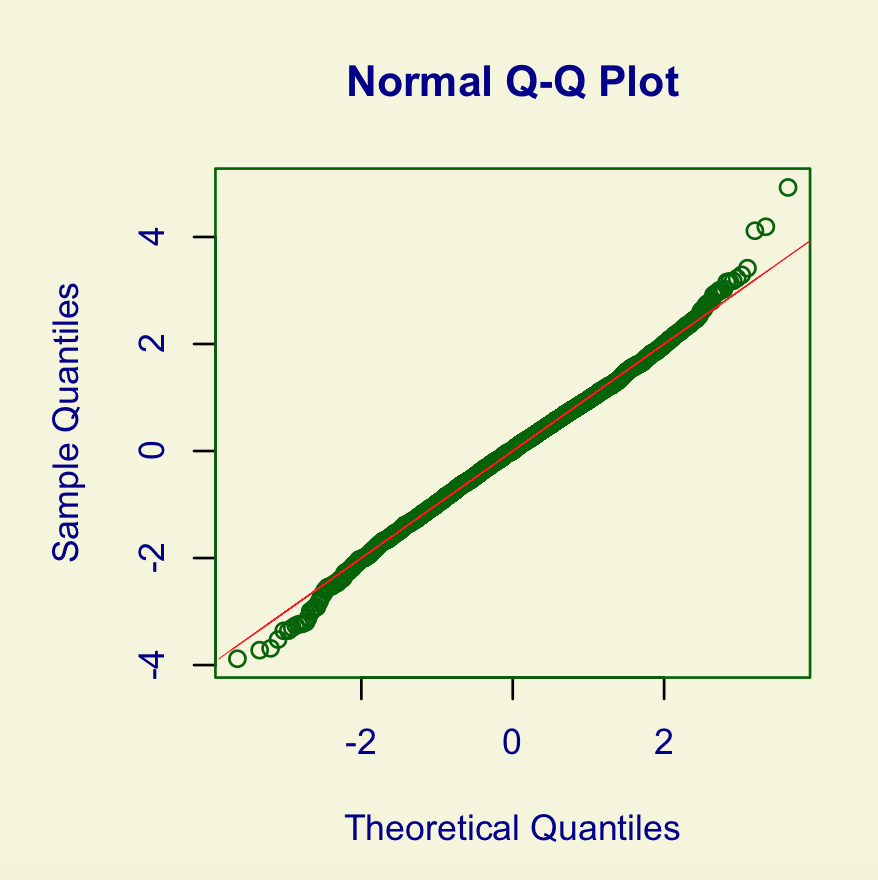
plot(TBE$V1[8:length(TBE$V1)], ewres[8:length(TBE$V1)], xlab="Date", ylab="Number of Consecutive Alerts")

abline(h=4, col=2)

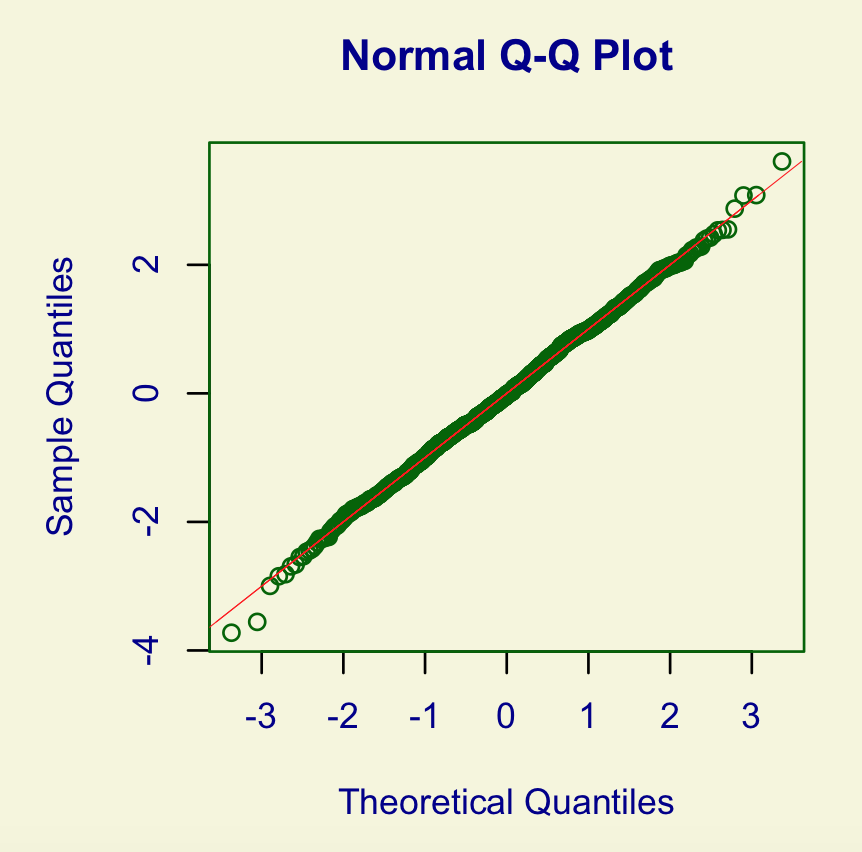
allops <- cbind(TBE,ewres)

alerts<-allops[allops$ewres >= 4,]

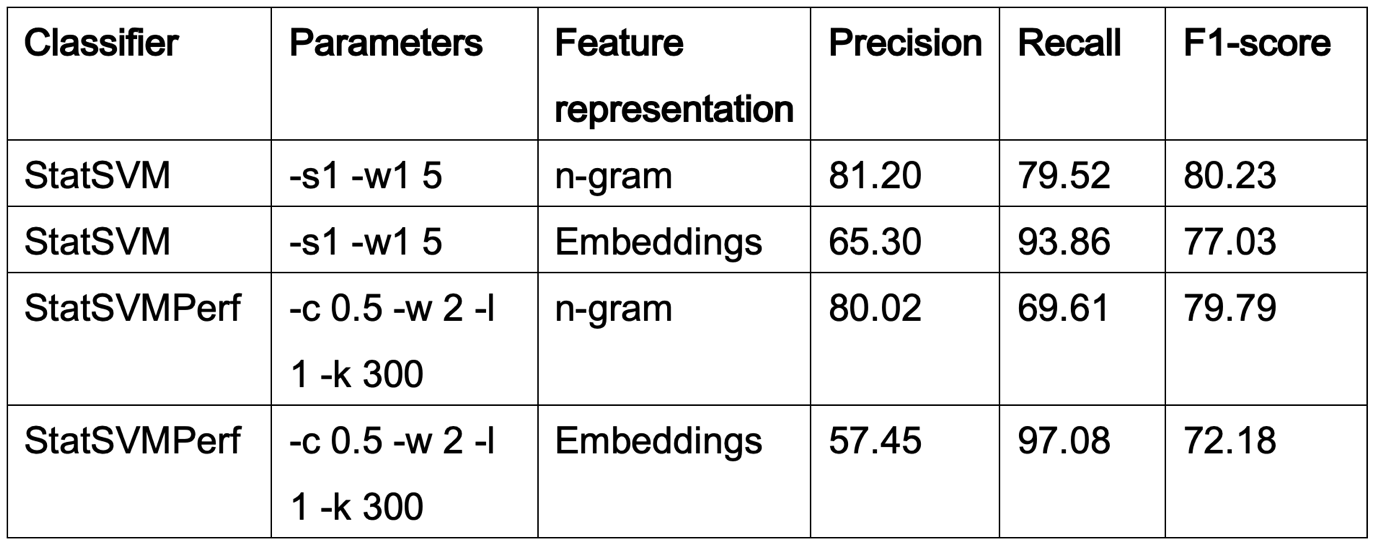
length(alerts$V1)



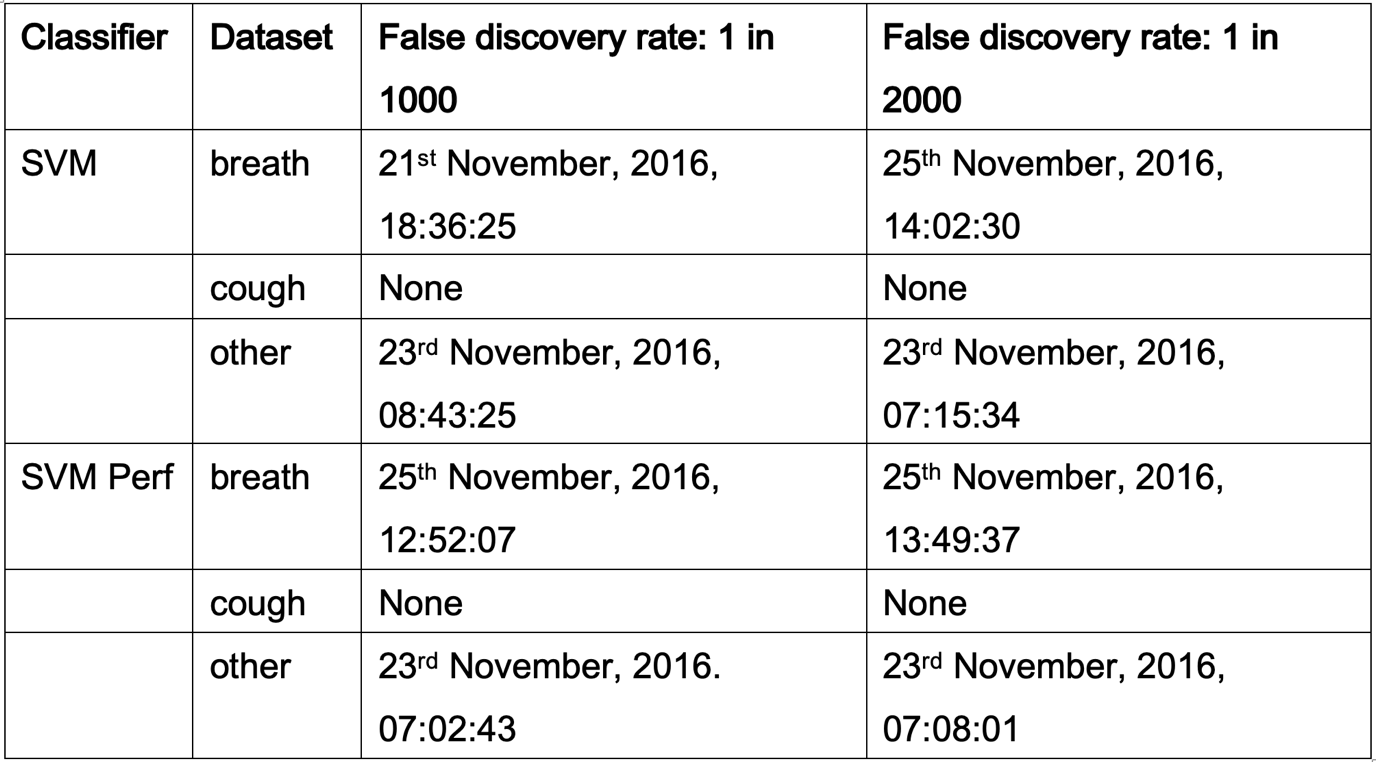
eFigure1a



eFigure1b



eTable1



eTable2