**ML Theoretical questions**

1. Accuracy is calculated as and is also a part of the performance's statistics as well as- F1 score, sensitivity, specificity, TP, FP, etc...

Similarly to the example presented in lecture 8, Integrating a large population samples can be can be misleading considering the accuracy only. If we take a 900 patients population-100 of them have AF and 800 are non-AF, we can get very imbalanced data so there will be poor learning even though the accuracy is 89%. It is common in medicine.   
To conclude, stats other than Accuracy provide better insights when classes are skewed.

**BP and BMI features**

Pros

* Lower complexity when taking less features into consideration- faster calculation time and less samples to analyze.
* These are two important features that connect both T1D and heart attacks so it may give a simple model with lower chances to overfitting

Cons

* There is a tradeoff between overfitting and bias, so using this kind of model may not give us enough data which would result with bias.
* There are many important features to take into consideration and these two may not be the most relevant in all the patient.
* May cause underfitting

**All of the features**

Pros

* Taking more important and relevant features into consideration
* Lower chances to get bias
* Still a relatively low number of features that provides us bigger information

Cons

* larger complexity and greater calculation time
* bigger chances for overfitting
* more samples are needed for the analysis and we may take less relevant parameters in count

we would prefer to take all of the features because taking the relatively small number of features into consideration, the differences in the bias are more significant than the chances getting overfitting in the second model. In that way, we would get more accurate model that would increase the chance to predict which patients are going to experience a heart attack.

1. Depending on the number of training sets (data)/features that you have, we can choose to use either logistic regression or support vector machine.

For a small number of features (1–1000) and intermediate number of training examples (10–10,000) logistic regression and linear SVM would give us the similar results.

The biopsies are really similar, and it is difficult to distinguish them from the human eye, or just by looking at the features. Thus, we can infer that the data is not linearly separable, but it may be linearly separable in higher dimensions. Therefore, in this case we will prefer to use a nonlinear SVM over logistic regression or a linear SVM. However, we should be careful because it my cause overfitting. To sum up, there is no one ideal model to use but in this particular case when we take into consideration data separation and the features and examples size, we would prefer a nonlinear SVM over the other options mentioned.

1. Both logistic regression and linear SVM are linear classifications models.   
   Logistic regression is a great model in a low number of dimensions and when the predictors don't suffice to give more than a probabilistic estimate of the response. SVMs do better when there's a higher number of dimensions, and especially for problems where the predictors certainly (or near-certainly) determine the responses. Additionally, SVM is a more geometric model, only considering points near the margin (support vectors) and trying to maximize them. Logistic regression considers all the points in the data set and focuses on maximizing probabilities. moreover, SVM is more complex than LR and allows us to make inclusion in addition to data separation. Also, SVM is more sensitive to overfitting due to its complexity. On the other hand, logistic regression is more sensitive to underfitting.

**Logistic regression hyperparameters**

λ

A hyper parameter that sets the penalization. The higher it is, the model will be pushed towards lower weights in order to minimize the cost function and there will be a larger emphasis on the regularization.

Choosing the learning rate is challenging, as a value too small may result in a long training process that may not converge, whereas a value too large may result in learning a sub-optimal set of weights too fast or an unstable training process.

**The order of the polynomial**

Regularization can also be performed in a regression task, where it is supposed to

achieve the optimal weights values of the chosen order of the polynomial model for instance.

**Penalty**

* **L1(Lasso)** - limits the size of the coefficients. Some coefficients can become zero and become eliminated
* **L2(Ridge)- regularization**adds a penalty that is equal to the square of the magnitude of coefficients.
* **Elastic nets** combine L1 & L2 methods

**Linear SVM hyperparameters**

C

C- It is a hypermeter in SVM to control error. The C parameter tells the SVM optimization how much you want to avoid misclassifying each training example.

C=1\λ

For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly (hard margin)

For small values of C, the optimization will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points (soft margins)

Gamma is used when we use the Gaussian RBF kernel. It is not relevant for the linear SVM.   
We would only need to use the C hyper parameter.