Question 1

Research a predictive modeling technique or machine learning algorithm of your choice that we did **not** study in the Data Analytics and Application course.

Give a detailed description of the model/algorithm by addressing all of the following questions:

- is the model/algorithm designed to address supervised or unsupervised tasks (or both)?
- how is the model/algorithm fitted to the data (e.g., which loss function/criterion is used?)
- what are the parameters of the model? which, if any, are interpretable parameters? which, if any, are instead tuning parameters?
- what is the relationship between each tuning parameter and the tendency of the model/algorithm to overfit or underfit? (e.g. are large values of the parameters associated with overfitting or underfitting?)
- if the model is designed for supervised tasks, can it be used to meaningfully interpret the association between each predictor and the response variables? if so, how?

Answer: When I talked to my friend about modeling, who works at Deloitte as a data scientist, she mentioned they are using GBM, Gradient Boosting Machines.
an example of such coding is

```
from sklearn.ensemble import GradientBoostingClassifier
gbm = GradientBoostingClassifier(n_estimators=100, learning_rate=0.1, max_depth=3, random_state=42)
gbm.fit(X_train, y_train)
predictions = gbm.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
```

i) is the model/algorithm designed to address supervised or unsupervised tasks (or both)?

This model is designed for supervised learning task, just like adaboost. Supervised problems involve a specific target (or response) variable, and the goal is to understand the relationship of the target to other feature variables. Boosting: using weighted combinations of “weak learners”. It sequentially combines weak models into a strong predictive model.

ii) how is the model/algorithm fitted to the data (e.g., which loss function/criterion is used?)

GBM improves its predictions step by step. In each step, it adds a new tree that tries to fix the mistakes made by the trees created before it. When it's learning, GBM tries to make its mistakes as small as possible. For tasks where it predicts a continuous number (like house prices), it tries to make the difference between the predicted and actual prices as small as possible. For tasks where it has to choose between categories, it tries to be as sure as possible about its choice. By focusing on the mistakes of the previous trees and combining them, GBM gets better and more accurate at making predictions.

iii) what are the parameters of the model? which, if any, are interpretable parameters? which, if any, are instead tuning parameters?

- Interpretable Parameters:

- Number of Trees: The total count of sequential trees to be modeled.

- Depth of Trees: The maximum depth of the individual trees.

- Learning Rate: Scales the contribution of each tree. A lower rate requires more trees but can yield a more generalized model.

- Tuning Parameters:

- Learning Rate (also interpretable): Determines how quickly the algorithm adapts to the problem. Smaller values make the model more robust to overfitting but require more trees.

- Subsample: The fraction of samples to be used for fitting the individual base learners. Less than 1.0 can lead to a reduction of variance and an increase in bias.

- Max Features: The number of features to consider when looking for the best split.

iv) what is the relationship between each tuning parameter and the tendency of the model/algorithm to overfit or underfit? (e.g. are large values of the parameters associated with overfitting or underfitting?)

- Number of Trees: bigger number increase the risk of overfitting. We should try to find the best estimator value by

```
gbt = GradientBoostingClassifier(random_state=42)
grid_search = GridSearchCV(estimator=gbt, param_grid=param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train, y_train)
best_estimator = grid_search.best_estimator_
```

- Depth of Trees: bigger number increase the risk of overfitting.

- Learning Rate: bigger number increase the risk of overfitting.

- Subsample: smaller sample size can precent the risk of overfitting.

- Max Features: bigger number increase the risk of overfitting.

v) if the model is designed for supervised tasks, can it be used to meaningfully interpret the association between each predictor and the response variables? if so, how?

Yes it can.

It is not as simple as linear regression. We can figure out how important each piece of information (feature) is by looking at how much it's used in all the decision-making trees and how much it helps make the predictions better. But understanding if a feature's impact is good or bad, or how exactly it affects the outcome (straightforwardly or in a more complex way), isn't always easy straight from the model. To get a clearer picture of how these pieces of information relate to the predictions, we can use special graphs called partial dependence plots after the model is ready.

Question 2 [3 points]

Based on your own professional experience, describe a real industry problem in which the model/algorithm that you discussed in Question 1 can be used. In particular, address all of the following questions:

- who are the stakeholders in this problem?
- how does solving this problem by means of the chosen model/algorithm generate value for them?
- what kind of data needs to be available to effectively solve this problem using the model/algorithm of choice?

Answer: Industry Problem: Predicting Hospital Readmission for Diabetes Patients

i) who are the stakeholders in this problem?

- Healthcare Providers: Doctors, nurses, and hospital administrators concerned with patient care quality.

- Patients: who are seeking effective treatment.

- Insurance Companies: Interested in minimizing costs associated with high rates of readmission.

- Policy Makers: Focused on improving healthcare outcomes and reducing healthcare costs at a systemic level.

ii) how does solving this problem by means of the chosen model/algorithm generate value for them?

Using GBM can provide value as for

- Healthcare Providers: can spot and concentrate on patients who needs readmission and give them special attention. This makes hospitals run more efficiently.

- Patients: Can be notified if they will need readmission and better prepare for after care.

- Insurance Companies: save money for who don't need readmission.

- Policy Makers: cost reduction.

iii) what kind of data needs to be available to effectively solve this problem using the model/algorithm of choice?

- Patient Data: Age, sex, race, and socio-economic status.

- Preconditions: Previous medical diagnosis and existing medications.

- Lab Results.

- destination data: readmission (yes or no). readmission timeframe and Length of stay.

By using GBM, we can better understand reasons of readmission for diabetic patients. This helps better planning and cost reduction.

Question 3 [2 points]

In Questions 1 and 2, our reasoning was backwards: first we selected an algorithm, and then we thought of a problem that suits it. In this question, we will re-think about these two steps in a more appropriate order. Given the problem of Question 2, and considering its context and the relevant stakeholders, what would be the most appropriate choice of model/algorithm to address it? Why? (This may or may not be the model/algorithm that you discussed in Question 1.)

Answer: For the hospital readmission for diabetes patients problem, we want to choose a model or algorithm that is accurate, interpretable, and can adjust new/changed variables. My friend told me that they use Random Forest a lot.

```
rf = RandomForestClassifier(n_estimators=100, max_depth=5, random_state=42)
rf.fit(X_train, y_train)
predictions = rf.predict(X_test)
accuracy = accuracy_score(y_test, predictions)
```

When we look into random forest, It is bagging with randomly selected features, It's a type of ensemble learning method, where multiple models (specifically decision trees) are combined to make more accurate and stable predictions.

Random Forest starts by creating multiple decision trees. Each tree is built from a random sample of the data, which can involve both random selection of instances (rows) and features (columns). This randomness helps in making the model more robust and **less prone to overfitting**.

For regression tasks, the forest's prediction is usually the average of all the decision trees' predictions. It can handle **non-linear relationship**, which is usually the case for healthcare data.

When building a tree, each time a split is considered, a random subset of the features is chosen as candidates. This ensures diversity among the trees and contributes to the robustness of the model. This helps us find variables that we **could have missed**.

It can be used for both **numerical and categorical data** and perform well on large datasets.

Random Forest is a go-to algorithm due to its simplicity, interpretability, effectiveness, and relatively low risk of overfitting, making it widely used across different domains.

Random Forest will be a more appropriate choice.