1.SVM(Support Vector Machine):

Advantages:

1. SVM works relatively well when there is a **clear margin** of **separation** between classes.
2. SVM is more **effective** in high **dimensional** spaces.
3. SVM is effective in cases where the **number of dimensions** is **greater** than the **number of samples**.
4. SVM is relatively **memory** efficient

Disadvantages:

1. SVM algorithm is **not suitable** for **large data sets**.
2. SVM **does not perform** very well when the data set has **more noise** i.e., target classes are overlapping.
3. In cases where the **number of features** for each data point exceeds the **number of training** data samples, the SVM will **underperform**.
4. As the support vector classifier works by putting data points, above and below the classifying **hyperplane** there is **no probabilistic** explanation for the classification.

Logistic classification:

Advantages:

1. Easy to **implement** and **interpret** yet **efficient** in training
2. The predicted parameters give **inference** about the importance of each feature

3. Performs well on **low-dimensional** data.
4. Very efficient when the dataset has features that are **linearly separable**.

Disadvantages:

1. **Overfits** on **high dimensional** data
2. **Non-linear** problems can't be solved with logistic regression since it has a **linear decision** surface
3. **Assumes linearity** between dependent and independent variables.
4. **Fails** to capture **complex** relationships.

Decision Tree Classification:

Advantages:

1. Compared to other algorithms decision trees requires **less effort** for data preparation during pre-processing.
2. A decision tree **does not require normalization** of data.
3. A decision tree **does not require scaling** of data as well.
4. **Missing values** in the data also do **NOT affect** the process of building a decision tree to any considerable extent.

Disadvantages:

1. A **small change** in the data can **cause** a **large change** in the structure of the decision tree causing instability.
2. For a Decision tree sometime **calculation** can go far **more complex** compared to other algorithms.

3. Decision tree often involves **higher time** to train the model.

4. Decision tree training is relatively **expensive** as the complexity and time has taken are more.

5. The Decision Tree algorithm is **inadequate** for applying regression and **predicting continuous values.**

Random Forest Classification

Advantages:

1. Random Forest works well with both **categorical and continuous variables**.

2. Random Forest can automatically **handle missing values**.

3. Random Forest is usually **robust to outliers** and can handle them automatically.

4. Random Forest algorithm is very **stable**. Even if a new data point is introduced in the dataset, the overall algorithm is not affected much since the new data may impact one tree, but it is very hard for it to impact all the trees.

5. Random Forest is comparatively **less impacted by noise**.

Disadvantages:

1. **Complexity:** Random Forest creates a **lot of trees** (unlike only one tree in case of decision tree) and combines their outputs. By default, it creates 100 trees in Python sklearn library. To do so, this algorithm

requires much **more computational power and resources**.

2. **Longer Training Period:** Random Forest require much **more time** to train as compared to decision trees as it generates a lot of trees (instead of one tree in case of decision tree) and makes **decision** on the **majority of votes**.

KNN-Classification

Advantages:

1. **No Training Period**- KNN modelling **does not include training period** as the data itself is a model which will be the reference for future prediction and because of this it is very time efficient in term of **improvising** for a **random modelling** on the available data.

2. **Easy Implementation**- KNN is very easy to implement as the only thing to be calculated is the distance between different points on the basis of data of different features and this **distance can easily be calculated** using distance formula such as- **Euclidian or Manhattan**

3. As there is no training period thus new data can be **added** at **any time** since it won't affect the model.

Disadvantages:

1. **Does not work well with large dataset** as calculating distances between each data instance would be very costly.

2. **Does not work well with high dimensionality** as this will complicate the distance calculating process to calculate distance for each dimension.

3. **Sensitive to noisy and missing data**

4. **Feature Scaling-** Data in all the dimension should be scaled (normalized and standardized) properly.

Naïve Bayes Classification:

Advantages:

1. This algorithm **works quickly** and can save a lot of time.
2. Naive Bayes is suitable for solving **multi-class prediction problems**.
3. If its assumption of the **independence** of features holds true, it can perform **better than other models** and requires much less training data.
4. Naive Bayes is better suited for **categorical input variables** than numerical variables.

Disadvantages:

1. Naive Bayes assumes **that all predictors** (or features) are **independent**, rarely happening in real life. This limits the applicability of this algorithm in real-world use cases.
2. This algorithm faces the '**zero-frequency problem**' where it assigns zero probability to a categorical variable whose category in the test data set wasn't available in the training dataset. It would be best if you used a smoothing technique to overcome this issue.
3. Its estimations **can be wrong in some cases**, so you shouldn't take its probability outputs very seriously.

AdaBoost Algorithm

Advantages:

1. It is **easy to use** as we do not have to do many **hyperparameters** tunning as compared to other algorithms.
2. Adaboost **increases** the **accuracy** of the weak machine learning models.
3. Adaboost has immunity from **overfitting of data** as it runs each model in a sequence and has a weight associated with them

Disadvantages:

AdaBoost uses a **progressively learning boosting technique**. Hence high-quality data is needed in examples of AdaBoost vs Random Forest. It is also **very sensitive to outliers** and **noise in data requiring the elimination** of these factors before using the data

XGBoost Algorithm:

Advantages:

1. **Regularization**: XGBoost has in-built L1 (Lasso Regression) and L2 (Ridge Regression) regularization **which prevents the model from overfitting**. That is why, XGBoost is also called regularized form of GBM (Gradient Boosting Machine).
2. **Handling Missing Values**: XGBoost has an in-built capability to **handle missing values**. When XGBoost encounters a missing value at a node, it tries both the left and right hand split and learns the way leading to higher loss for each node. It then does the same when working on the testing data.
3. **Parallel Processing**: XGBoost utilizes the **power of parallel processing** and that is why it is much **faster than GBM**. It uses multiple CPU cores to execute the model.

Disadvantages:

1. **Overfitting**: Overfitting is likely to occur in xgboost if xgboost parameters are not tuned properly.
2. **Training time**: Training time is pretty high for larger dataset, if you compare against catboost/lightgbm.

Lightgbm algorithm:

Advantages:

1.Faster training speed and higher efficiency: Light GBM uses a histogram-based algorithm i.e it buckets continuous feature values into discrete bins which fasten the training procedure.

2. Lower memory usage: Replaces continuous values to discrete bins which results in lower memory usage.

3.Better accuracy than any other boosting algorithm: It produces much more complex trees by following leaf wise split approach rather than a level-wise approach which is the main factor in achieving higher accuracy.

4.Compatibility with Large Datasets: It is capable of performing equally well with large datasets with a significant reduction in training time as compared to XGBoost.

Disadvantages:

1.Overfitting: Light GBM split the tree leaf-wise which can lead to overfitting as it produces much complex trees.

2.Compatibility with Datasets: Light GBM is sensitive to overfitting and thus can easily overfit small data.