**Random Forest**

Real world applications: Predict parts failures in manufacturing – Data mining

**Pros**:

1. They perform well even in the presence of a large number of features and a small number of observations.
2. Quick Prediction/Training Speed
3. Robust to Outliers
4. Ability to measure variable importance

**Cons:**

1. Model interpretability: Random forest models are not all that interpretable; they are like black boxes.
2. For very large data sets, the size of the trees can take up a lot of memory.
3. It can tend to overfit, but that can be mitigated by tuning the hyperparameters
4. Does not perform well when there is a diagonal decision boundary as it will take an ordinary random forest model many splits to describe that diagonal boundary.

**Why this model?**

Random Forest is very fast to train, and Since the data set is large (36177 rows for training data) having a model that is fast to train is a huge advantage, especially when using random search or grid search to find the optimal hyperparameters. Fast training combined with being okay with handling large number of features makes Random forest a good candidate

**References**:

<https://hackernoon.com/choosing-the-right-machine-learning-algorithm-68126944ce1f>

<https://towardsdatascience.com/why-random-forest-is-my-favorite-machine-learning-model-b97651fa3706>

<https://stats.stackexchange.com/questions/112148/when-to-avoid-random-forest>

**Logistic Regression**

Real world applications: Trauma and Injury Severity Score (TRISS), which is widely used to predict mortality in injured patient

**Pros**:

1. does not require too many computational resources
2. doesn’t require input features to be scaled
3. doesn’t require any tuning

**Cons**:

1. can’t solve non-linear problems with logistic regression since its decision surface is linear.
2. Logistic regression will not perform well with independent variables that are not correlated to the target variable and are very similar or correlated to each other.
3. Logistic Regression is also not one of the most powerful algorithms out there and can be easily outperformed by more complex ones.

Why this model:

Since we have a single dependent variable and it is also binary, Logistic regression would be a good candidate, it also has the analytical advantage of providing us with the probability of a certain event happening, so not just the binary outcome

References:

<https://www.quora.com/What-are-the-pros-and-cons-of-using-logistic-regression-with-one-binary-outcome-and-several-binary-predictors>

<https://towardsdatascience.com/real-world-implementation-of-logistic-regression-5136cefb8125>

**Support Vector Machines**

Real world applications: Face detection and classification of images

**Pros**:

1. good in a high-dimensional space
2. if the problem is not linearly separable, we can use an SVM with a nonlinear kernel
3. The hyperplane is affected by only the support vectors thus outliers have less impact.
4. good theoretical guarantees regarding overfitting
5. SVM is suited for extreme case binary classification.

**cons**:

1. For larger dataset, it requires a large amount of time to process.
2. Does not perform well in case of overlapped classes.
3. Selecting the appropriate kernel function can be tricky.

**Why this model?**

Looking at our data, its mostly categorical. after using one hot encoding it became binary data, and SVM is suited for that kind of data. Also there is a possibility that our data is separable, but its hard to determine that due to the high dimensionality of the data, so SVM would be a good option to test for that reason. If the data is separable, SVM will perform very well

References:

<https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589>

<https://data-flair.training/blogs/applications-of-svm/>

**Adaboost**

real world application: product recommendation for online shopping

**pros**:

1. very flexible, can be combined with any other machine learning classifier to boost its performance
2. Very robust to overfitting
3. Performs well on large data sets

**Cons**:

1. vulnerable to uniform noise.

**Why use Adaboost:**

Adaboost iteratively corrects the mistakes of the weak classifier and improves accuracy by combining weak learners, meaning we can select a classifier suitable for our data and boost its results. We can use a decision tree as our classifier without worrying too much about overfitting, its also applicable for continuous and categorical inputs

**References**:

1. <https://www.ubuntupit.com/machine-learning-algorithms-for-both-newbies-and-professionals/>
2. <https://www.educba.com/adaboost-algorithm/>
3. <https://stats.stackexchange.com/questions/20622/is-adaboost-less-or-more-prone-to-overfitting>