# Optum Data scientist Interview Task

**Reason for choosing XGBoost as final Algorithm:**

Upon trying multiple machine learning algorithms, I finalized XGBoost. Below is the explanation for choosing XGBoost over the other models.

After performing data analysis, I understood, it is a supervised and binary classification problem statement because in the dataset has a target variable is provided. I need to predict the output whether it is anomaly or not an anomaly. Before finalizing an algorithm, I tried different algorithms which are best suitable for binary classification i.e Logistic regression, Support vector machines and Xgboost. We can still try more algorithms.

Initially when I was trying to build a model, for each model, I was receiving test accuracy more than train accuracy, which is not an ideal scenario. It may be due to the difference in underlying distribution. So, then I applied stratification and also split data set into train and test in the proportions of 0.63 : 0.37 to overcome it.

## Logistic regression model:

Logistic regression is traditional machine learning algorithm for binary classification. It uses regression under the hood and predicts the probability of a class using Sigmoid function. It is performing good at classifying 'Anomaly-No' class, but struggling to classify 'Anomaly-Yes' due to the imbalance in the target feature.

Accuracies:

Train accuracy: 0.80; Test accuracy: 0.79

## Support vector machines model:

Support vector machines also called as SVMs. It is also a popular algorithm for classification. SVMs are suitable when dataset is small. SVM algorithm tries to find a hyperplane that classifies the data points in N-dimensional space. The dimension is of hyperplane depends on number of input features in the dataset. If number of features is two then the hyperplane is just a line. SVM kernel is a function that transforms the data from low dimensional to higher dimensional space. It converts non-linear separable data to linear separable data in higher dimension. Some of SVMs kernel functions are linear, RBF, Polynomial and sigmoid kernel.

I tried building SVM model using linear and RBF kernel. SVM with linear kernel is not performing well. RBF kernel improved the accuracy but the recall and precision scores are not as expected.

**SVM using Linear Kerne Accuracies:**

Train accuracy: 0.74; Test accuracy: 0.69

**SVM using RBF Kernel Accuracies:**

Train accuracy: 0.80; Test accuracy: 0.79

None of the above algorithms are not giving desired results. We can improve the models performance by doing hyper parameter tuning and treating imbalance in the data set using up sampling or SMOTE techniques.

## XGBoost (Extreme Gradient Boosting) Model:

Extreme gradient boosting is a decision tree-based ensemble method. In Xgboost decision trees are created sequentially. XGboost improved the model accuracy and also improved the recall, precision and f1 scores. The model is overfitting, it is due to the small data set.

**Accuracies:**

Train accuracy: 1.0; Test accuracy; 1.0

We can fix the overfitting condition using regularization technique.

The reason for choosing XGboost as the final model, despite the fact that it is overfitting, because the dataset is too small and it has the same pattern behaviour when predicting labels as we discovered during our initial data analysis.

**Precision and Recall scores for each model**

| MODEL | CLASS | PRECISION | RECALL |
|---|---|---|---|
| Logistic Regression | 0 | 0.82 | 0.9 |
| | 1 | 0.74 | 0.59 |
| SVM (Linear Kernel) | 0 | 0.84 | 0.67 |
| | 1 | 0.53 | 0.75 |
| SVM (RBF Kernel) | 0 | 0.82 | 0.9 |
| | 1 | 0.74 | 0.66 |
| XGBoost | 0 | 1 | 1 |
| | 1 | 1 | 1 |

# XGBoost

XGBoost comes under the category of Boosting techniques in Ensemble Learning. In boosting technique, the errors made by previous model are tried to be corrected by the next model by adding some weights to the models.

Most of the times XGBoost outperforms the other ensembled algorithms due to its speed and performance.

**Features of XGBoost:**

- XGBoost is used in both regression and classification problems
- It provides regularizations which helps in reducing overfitting
- Auto tree pruning
- Can handle missing values
- Takes care of missing values for some extent
- Supports parallel processing
- Cache optimization
- Efficient memory management when data size is more than RAM.

**How XGBoost works:**

XGBoost for classification calculates the probabilities of outcome. Initially XGBoost creates a base model to calculate the residuals. To do this it uses probability value 0.5 and subtract the output value with the 0.5 then gets the residual. Once it gets the residuals it will construct a Decision tree. These residuals are the initial predictions.

In order to build a tree **similarity score** of the residuals is calculated (It is considered as the root). Below is the formula for similarity score.

$$\frac{(\sum \text{Residual}_i)^2}{\sum [\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)] + \lambda}$$

Here, Lambda is the regularization parameter.

XGBoost selects a feature from the data set and start splitting the data (Residuals). Similarity scores of the right and left side split is calculated. Then **Gain** of the overall split is calculated using below formula.

$$\text{Gain} = \text{Left}_{\text{Similarity}} + \text{Right}_{\text{Similarity}} - \text{Root}_{\text{Similarity}}$$

In the same way similarity scores and gain for each feature is calculated. Whichever is having the highest gain will be selected to split the tree. XGBoost has a threshold for

minimum number of residuals in each leaf. This is determined by calculating something called **Cover value.** Cover is defined as the denominator of the Similarity score minus lambda.

$$\text{Similarity} = \frac{(\sum \text{Residual}_i)^2}{\sum [\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)] + \lambda}$$

$$\text{Cover} = \sum [\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)]$$

Lambda reduces the similarity score and that lower similarity score result in lower values for Gain. When the difference between Gain and Gamma is negative XGBoost prune the tree (Gamma is user defined). That means lower value of Gamma will result in a negative difference cause to prune the tree.

$$\text{Gain} - \gamma = \begin{cases} \text{If positive, then do not prune.} \\ \text{If negative, then prune.} \end{cases}$$

After building a complete tree for classification, the output values for each leaf are calculated using below formula.

$$\frac{(\sum \text{Residual}_i)}{\sum [\text{Previous Probability}_i \times (1 - \text{Previous Probability}_i)] + \lambda}$$

Lambda, the regularization parameter > 0, reduces the prediction sensitivity.

XGBoost for classification makes new predictions by starting with initial prediction (residual) using probability. It will get converted into log(odds) value.

$$\frac{p}{1 - p} = \text{odds}$$

$$\log\left(\frac{p}{1 - p}\right) = \log(\text{odds})$$

XGboost then calculates the log(odds) prediction.

**[ Log(odds) Prediction = Log(odds) of initial prediction + (Learning rate * output value) ]**

To convert the log(odds) value into a probability, XGBoost uses **Logistic Function** then it will get new predicted probabilities.

$$\text{Probability} = \frac{e^{\log(\text{odds})}}{1 + e^{\log(\text{odds})}}$$

These are the new residuals. When we compare new residual to previous residual it is smaller. So, the model has taken a small step in the right direction. The new predictions for the remaining observations have smaller Residuals than before.

Now, that the model has new residuals, it will build a second tree that is fit to the new residuals. The new tree predicts the even smaller residuals. Likewise, XGboost keep on building the trees until the Residuals are very small, or it has reached the maximum number of trees.

In summary, when building XGBoost trees for classification the model calculates the Similarity scores and Gain to determine how to split the data and it will prune the tree by calculating the difference between Gain and Gamma. The it calculates the output values for the leaves. Lambda is a regularization parameter when lambda > 0, it results in more pruning, by shrinking the Similarity scores, and smaller output values for the leaves.

Thank you.