**PROJECT REPORT**

**ON**

**CREDIT CARD APPROVAL PREDICTION**



Submitted by-

**Team Members**

Krishna Ravichandran (001001577)

Tharoon Kumar Viswanathan(001585510)

Lokesh Balaji Parameswaran(001562875)

**Abstract**

This report consists of the final findings of the Supervised Machine Learning algorithms used for the prediction of the Credit card Approval based on the various features.

The dataset used for the project has been taken from Kaggle. The data is then cleaned as per the requirement and then various models were built in Jupyter Notebook (Anaconda 3) to examine the models’ performance on certain parameters.

After a preliminary study of the available algorithms and data review, it became apparent that the problem fell under non-linear Classification category. The study focuses on various algorithms by using classifiers like- XG Boost Classification, Decision Tree Classification, K Nearest Neighbour Classification, Random Forest Classification and Deep Neural Network.

**Table of Contents**

* Introduction
  + Background
  + Task
  + Goal
* Methodologies and Algorithms
  + Data Review
  + Software and Libraries used
  + Data Cleaning
  + Feature Selection
  + Exploratory Data Analysis
  + Methods used
    - XG Boost Classification with SMOTE
    - Decision Tree
    - Rndom Forest Classification
    - K Nearest Neighbour Classification with hyper parameter tuning
    - Deep Nural network
* Description of Dataset
* ML Model implementation
* Results and Analysis
* Conclusion
* References

**Introduction**

**Background**

Credit score cards are a common risk control method in the financial industry. It uses personal information and data submitted by credit card applicants to predict the probability of future defaults and credit card borrowings. The bank is able to decide whether to issue a credit card to the applicant. Credit scores can objectively quantify the magnitude of risk.

Generally speaking, credit score cards are based on historical data. Once encountering large economic fluctuations. Past models may lose their original predictive power. Logistic model is a common method for credit scoring. Because Logistic is suitable for binary classification tasks and can calculate the coefficients of each feature. In order to facilitate understanding and operation, the score card will multiply the logistic regression coefficient by a certain value (such as 100) and round it.

At present, with the development of machine learning algorithms. More predictive methods such as Boosting, Random Forest, and Support Vector Machines have been introduced into credit card scoring. However, these methods often do not have good transparency. It may be difficult to provide customers and regulators with a reason for rejection or acceptance.

**Task**

Build a machine learning model to predict if an applicant is 'good' or 'bad' client, different from other tasks, the definition of 'good' or 'bad' is not given. You should use some techique, such as vintage analysis to construct you label. Also, unbalance data problem is a big problem in this task

**Goal**

Our goal is to determine whether a every applicant is 'good' or 'bad'. We will be achieving this by building a machine learning model to learn the features and tune it to predict the target (STATUS) variable.

* Using supervised machine learning methods to predict the STATUS value of an asteroid as Y/N
* To determine the most efficient algorithms with the highest accuracy score for the given Dataset
* Visualizing the performance of the models using confusion matrix and ROC curves.

**Methodology & Algorithm**

**Data Review**

The dataset contains , two datasets records of all the applicant and credit history with balance till current month, it has 438558 records (70 MB size) and contains 20 columns.

**Software and Libraries used**

The dataset is downloaded from Kaggle. Jupyter notebook is used for the coding aling with several Libraries

That are-

* Numpy
* Pandas
* Matplotlib
* Seaborn
* Metrics
* Scikitlearn Tree
* XG Boost Classifier
* DecisionTreeClassifier
* train\_test\_split
* sklearn.ensemble
* RandomForestClassifier
* plotly.graph\_objects
* sklearn.preprocessing
* Get\_dummies
* RepeatedStratifiedKFold
* Pydotplus
* Graphviz
* F1\_score, ROC\_curve, accuracy\_score
* Sqrt
* Mean\_squared\_error

And several others for classification models.

**Data Cleaning**

Data provided was heterogenous with a couple columns containing missing values. Many rows had Null values which can degrade the model’s performance, hence we need to take care of all the rows with Null values. We used the mode values of the features to fill in the missing fields instead of leaving them Null and removed the rows that had null values after filling the mean values.

**Feature Selection**

#### The data originally had 20 columns, this can lead to too much noise and degrade the performance of the model. To avoid this noise we dropped and retained selected features based on the correlation matrix and comparing how it affects the target variable. Also, removed duplicate columns which were highly correlated to each other.

**Exploratory Data Analysis**

After cleaning the data and sorting it we have done feature selection on it using various pandas and matplotlib commands. We have shown a histogram of all relevant columns using hist() and visualized the percentage of features and also printed a bar chart of STATUS count in the dataset. This visuvalisation was used to identify the imbalance in the dataset.

**Models Used**

The Dependent Target Variable is STATUS which contains a boolean value of Y or N

As the Target Variable is a discrete class value, the prediction model used is **Classification**.

The dataset with its matrix of features (independent variables) is trained on various Classification models to predict the class value of the dependent target variable.

Different types of classifiers used in the project are-

* XG Boost Classification using SMOTE
* Decision Tree Classification
* K Nearest Neighbour Classification with Hyperparameter tuning
* Random Forest Classification
* Deep Neural Network

We have also tried using the above models after tuning the hyperparamaeters and after oversampling the minority class.

1. **XG Boost Classification with SMOTE**

XGBoost  classifier is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. However, when it comes to small-to-medium structured/tabular data, decision tree based algorithms are considered best-in-class. The classifier was pretty accurate with the prediction of the target variable for our data set. SMOTE (Eynthetic samples) was used to overcome the oversampling of the target variable.

1. **K Nearest Neighbour Classification with Hyper Parameter Tuning**

K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. Once the model is trained Hyper parameter tuning were done to improve the learning of the model.

1. **Deep Neural Network**

Deep learning [algorithms](https://www.datarobot.com/wiki/algorithm/) run data through several “layers” of [neural network](https://www.datarobot.com/wiki/neural-network/) algorithms, each of which passes a simplified representation of the data to the next layer. Deep learning algorithms learn progressively more about the image as it goes through each neural network layer. Early layers learn how to detect low-level features like edges, and subsequent layers combine features from earlier layers into a more holistic representation.This is the most advanced ML model used in training the dataset

1. **Decision Tree Classification**

Decision tree is highly useful in classification problems where the total number of features and rows is high. A decision tree is represented as upside down where its root is at the top of the tree then it splits into branches and when it cannot further split then the end branch is called as decision. Growing a decision tree requires to choose features and conditions to select optimal tree which has maximum prediction. The tree is grown arbitrary.

1. **Random Forest Classification**

Random forest is at par with decision tree in terms of getting result both have given satisfactory results. The random forest is a flexible algorithm which is easy to use and takes very less time when compared to logistic regression. In decision tree only one tree is made but in random forest, our algorithm randomly creates a specified number of decision trees. And chooses the tree which is best for our model.This model gave us the second best result with respect to the F1 score the a tie along with the decision tree model for the accuracy score.

**Visualization**

Visualization is done at classification models as well as in EDA. Where bar graph, line graphs, scatter plot, Roc Curves, confusion matrices etc. are used to represent data as well as data frames have been used. The following libraries were used as well: Matplotlib, Seaborn and Plotly.graph.

**Description of Dataset**

The Credit card Approval Prediction Dataset contains all the information contains about an applicant and credit status to date.

The STATUS column (Target variable column) tells us if an applicant is 'good' or 'bad' client for availinga credit card or not and uses Boolean value (Y or N). It uses over due days and loan months.

The dataset consists of two .csv file which is 70 MB and has 438558 records and contains 21 columns.

Of which:

* 9 attributes are String,
* 10 attributes are Int,
* 2 attributes are Decimal

List of Attributes:

|  |  |  |
| --- | --- | --- |
| **Application\_record.csv** |  |  |
| **Feature name** | **Explanation** | **Remarks** |
| ID | Client number |  |
| CODE\_GENDER | Gender |  |
| FLAG\_OWN\_CAR | Is there a car |  |
| FLAG\_OWN\_REALTY | Is there a property |  |
| CNT\_CHILDREN | Number of children |  |
| AMT\_INCOME\_TOTAL | Annual income |  |
| NAME\_INCOME\_TYPE | Income category |  |
| NAME\_EDUCATION\_TYPE | Education level |  |
| NAME\_FAMILY\_STATUS | Marital status |  |
| NAME\_HOUSING\_TYPE | Way of living |  |
| DAYS\_BIRTH | Birthday | Count backwards from current day (0), -1 means yesterday |
| DAYS\_EMPLOYED | Start date of employment | Count backwards from current day(0). If positive, it means the person currently unemployed. |
| FLAG\_MOBIL | Is there a mobile phone |  |
| FLAG\_WORK\_PHONE | Is there a work phone |  |
| FLAG\_PHONE | Is there a phone |  |
| FLAG\_EMAIL | Is there an email |  |
| OCCUPATION\_TYPE | Occupation |  |
| CNT\_FAM\_MEMBERS | Family size |  |
|  |  |  |
| **Credit\_record.csv** |  |  |
| **Feature name** | **Explanation** | **Remarks** |
| ID | Client number |  |
| MONTHS\_BALANCE | Record month | The month of the extracted data is the starting point, backwards, 0 is the current month, -1 is the previous month, and so on |
| STATUS | Status | 0: 1-29 days past due 1: 30-59 days past due 2: 60-89 days overdue 3: 90-119 days overdue 4: 120-149 days overdue 5: Overdue or bad debts, write-offs for more than 150 days C: paid off that month X: No loan for the month |

**Data Source**

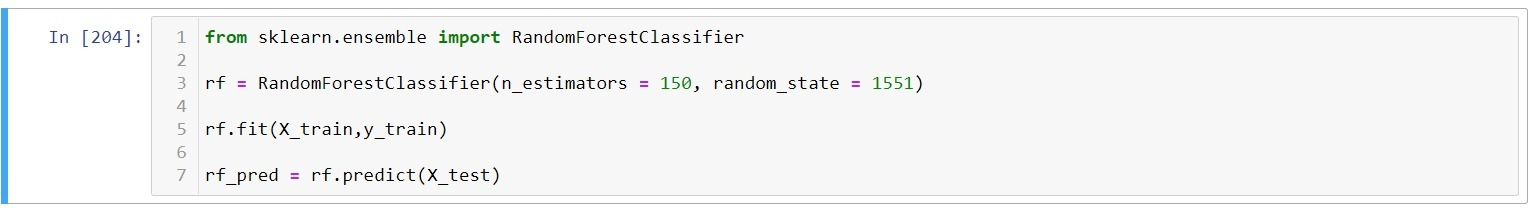
The dataset has been taken from Kaggle:

Credit Card Approval Prediction Dataset

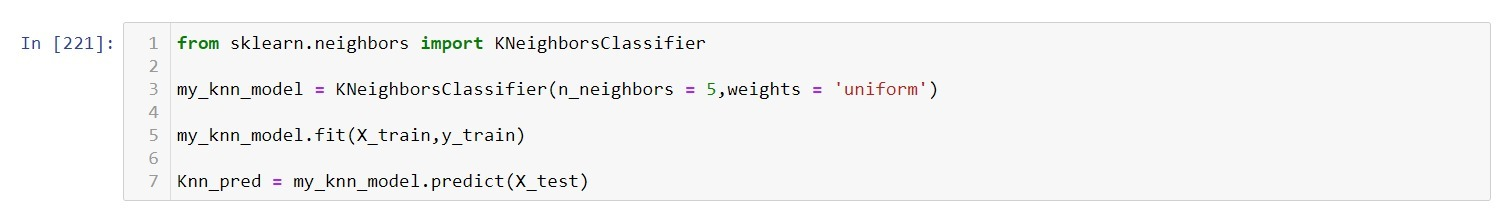
<https://www.kaggle.com/rikdifos/credit-card-approval-prediction>

**Models for Predicting the Credit card Approval Prediction**

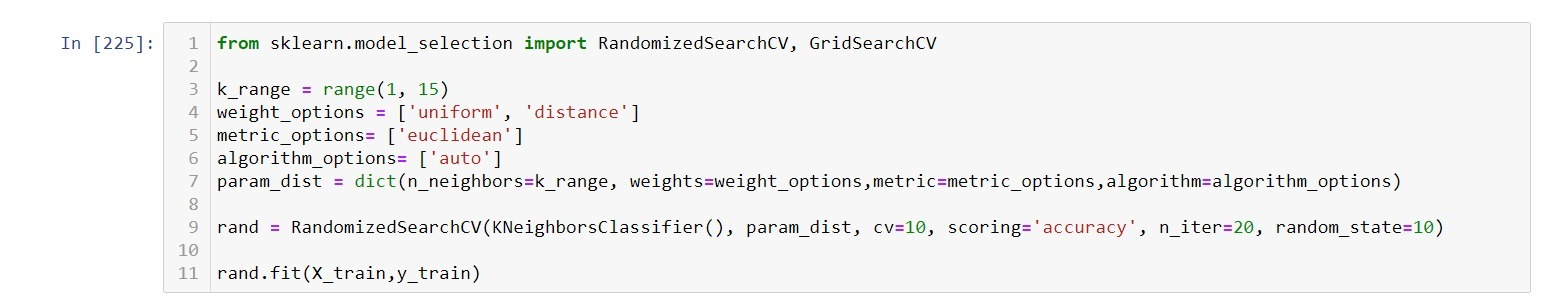
**1.Random Forest:**



**2.K Nearest Neighbor:**



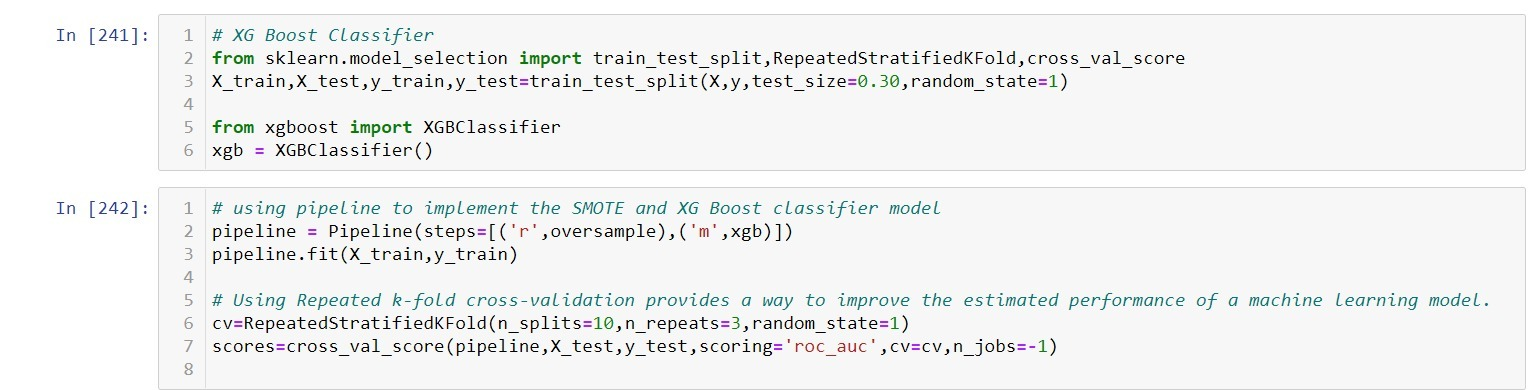
**3.K Nearest Neighbor with Hyper parameter tuning:**



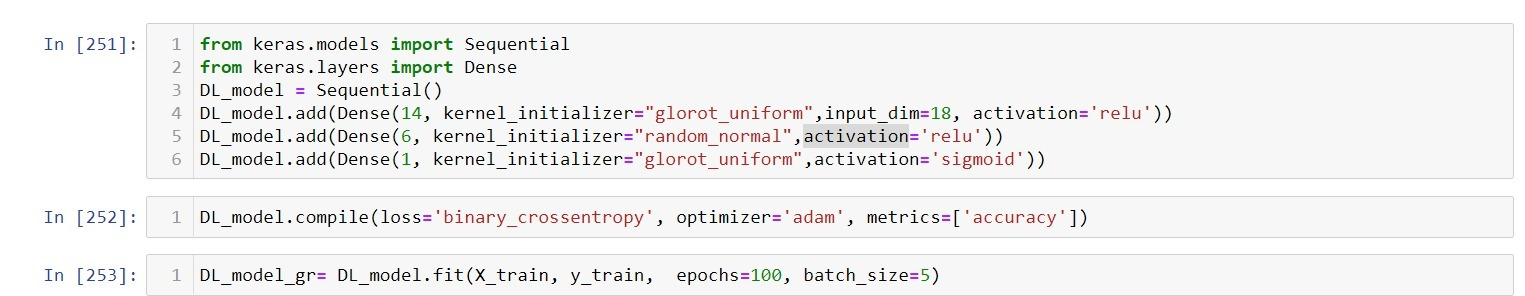
**4.Decision Tree Classifier:**



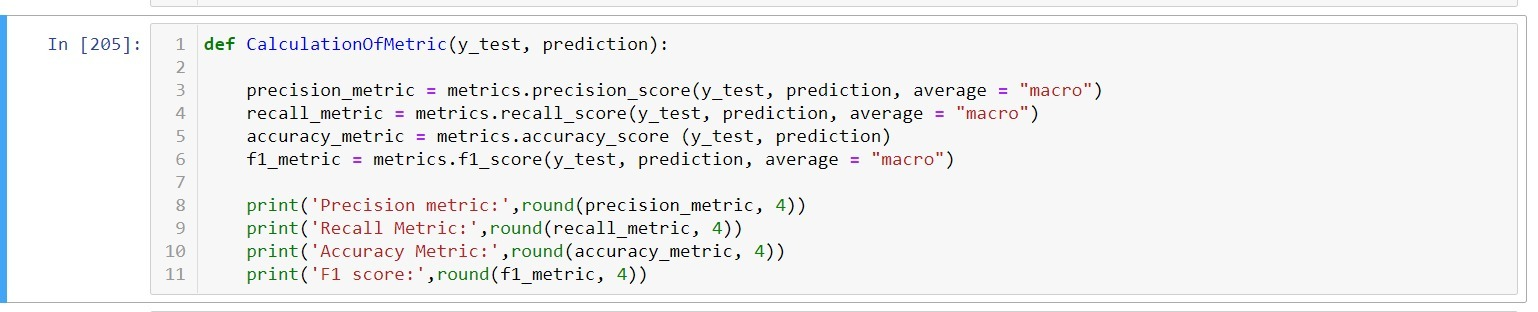
**5. XG Boost Classifier:**



**5. DNN Model using Keras:**



**Results And Analysis**

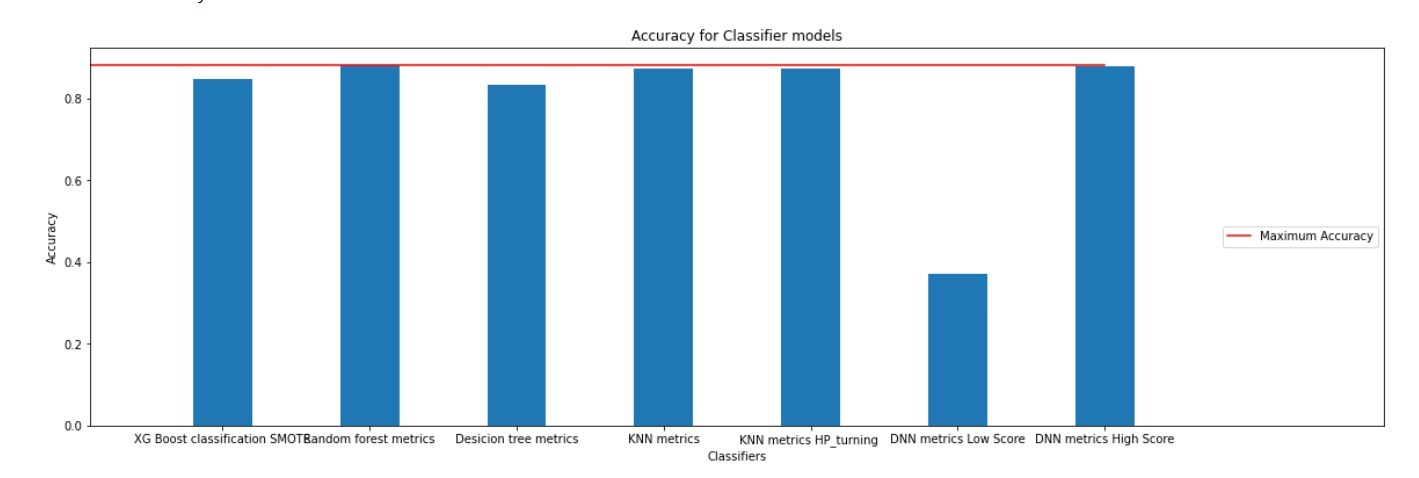


**Performance Metric Measures**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Precision** | **Recall** | **Accuracy** | **F1 Score** |
| XG Boost Classification | 0.6268 | 0.6206 | 0.8429 | 0.6235 |
| Decision Tree Classifictaion | 0.6199 | 0.6329 | 0.8311 | 0.6257 |
| K Nearest Neighbour Classification | 0.6738 | 0.5903 | 0.8722 | 0.6115 |
| KNN with hyperparameter tuning | 0.6216 | 0.5277 | 0.8712 | 0.5263 |
| Random Forest Classifictaion | 0.71 | 0.6172 | 0.8801 | 0.6439 |
| Deep Neural Network | 0.4391 | 0.5 | 0.8782 | 0.9351 |

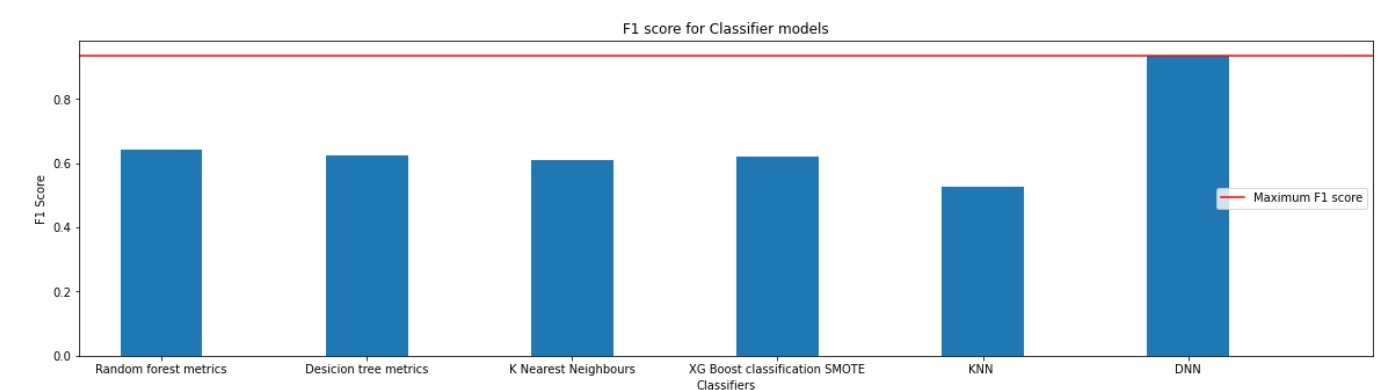
**Accuracy Score:**

* Bar graph of accuracy scores for all models



**F1 Score:**

* Visualizing F1 scores of all models as a bar graph



**Conclusion:**

For this Dataset we have implemented the 5 types of models, from our observation through the performance metrics we were able to find out the best model among the five used. In our case Random Forest Classification has the best accuracy of (0.881) after implementing hyperparamter tuning and DNN model has the best F1 Score of (0.9351).

**References:**

* [**https://www.digitalocean.com/community/tutorials/how-to-build-a-deep-learning-model-to-predict-employee-retention-using-keras-and-tensorflow**](https://www.digitalocean.com/community/tutorials/how-to-build-a-deep-learning-model-to-predict-employee-retention-using-keras-and-tensorflow)
* [**https://stackoverflow.com/questions/50920908/get-confusion-matrix-from-a-keras-multiclass-model**](https://stackoverflow.com/questions/50920908/get-confusion-matrix-from-a-keras-multiclass-model)
* [**https://de.mathworks.com/help/matlab/ref/colormap.html**](https://de.mathworks.com/help/matlab/ref/colormap.html)
* [**https://machinelearningmastery.com/evaluate-performance-deep-learning-models-keras/**](https://machinelearningmastery.com/evaluate-performance-deep-learning-models-keras/)
* [**https://machinelearningmastery.com/how-to-calculate-precision-recall-f1-and-more-for-deep-learning-models/**](https://machinelearningmastery.com/how-to-calculate-precision-recall-f1-and-more-for-deep-learning-models/)
* [**https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d#:~:text=What%20is%20XGBoost%3F,all%20other%20algorithms%20or%20frameworks**](https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d)