Logo, company name

Description automatically generated

**Design and Implementation of Analytics System**

**Effective Advertising Strategy**

**CENSORED**

Raditya Fahritama

**School of Graduate Professional Studies**

Data Analaytics

DAAN 888 – Design and Implementation of Analytics System

Fall, 2022

# Document Control

## Work carried out by:

|  |  |  |
| --- | --- | --- |
| **Name** | **Email Address** | **Task description** |
| **CENSORED** | **CENSORED** |  |
| Raditya Fahritama | rkf5230@psu.edu |  |
|  |  |  |

## Revision Sheet

|  |  |  |
| --- | --- | --- |
| **Release No.** | **Date** | **Revision Description** |
| 1.0 | 09-04-2022 | Initial Draft |
| 2.0 | 09-18-2022 | Data Collection and Data Source Information |
| 3.0 | 10-15-2022 | Data cleaning |
| 4.0 | 11-02-2022 | Variable selection and Transformation |
| 5.0 | 11-13-2022 | Data Modelling |
|  |  |  |
|  |  |  |

[Document Control 1](#_Toc763423140)

[Work carried out by: 1](#_Toc1234281342)

[Revision Sheet 1](#_Toc877774100)

[Academic Integrity 3](#_Toc1887155537)

[How Academic Integrity Violations Are Handled 4](#_Toc568093233)

[For More Information on Academic Integrity at Penn State 4](#_Toc1622729378)

[Week 2 Predictive / Descriptive Analytics System Group-Based Assignment 1](#_Toc659626360)

[WEEK 4 Predictive/Descriptive Analytics System Group-Based Assignment 3](#_Toc824391130)

[WEEK 6 Predictive Analytics System Group-Based Assignment 4](#_Toc1205267827)

[WEEK 8 Predictive/descriptive Analytics System Group-Based Assignment 5](#_Toc428908938)

[WEEK 10 Predictive/descriptive Analytics System Group-Based Assignment 6](#_Toc1523047924)

[WEEK 12 Predictive / Descriptive Analytics System Group-Based Assignment 7](#_Toc525671129)

[WEEK 13 Predictive/descriptive Analytics System Group-Based Assignment 8](#_Toc1053416539)

[WEEK 14 Predictive / Descriptive Analytics System Group-Based Assignment 9](#_Toc1855548708)

[References. 10](#_Toc1869801335)

**General Guidelines**

1. To complete all the homework assignments for this course please use this template document.
2. Each assignment has to be submitted by Sunday 11:59 PM EST.
3. Each figure should be followed by a brief description about the figure.
4. The figures can be hand drawn and scanned in some circumstances, but the hand drawn figure should be clear and legible to obtain full credits. Unclear hand drawn figures will receive partial credits. For constructing figures and diagrams it is advised to use tools.
5. Figures and tables should have appropriate captions. For documenting and referencing styles please follow the APA or MLA writing style.
6. Please make sure that you provide a reference section.
7. Any material text or figure taken from books, journals or Internet should be referenced. If you have a sentence or a figure that does not belong (authorship) to you, they need to be clearly referenced. If you fail to do so your report will be considered as a case for plagiarism. It is your duty to make sure that your report is free from any activity related to plagiarism. In case you are suspected of attempting plagiarism then you will be responsible for the cause. The penalty for plagiarism will be a “0” awarded to your report. So, it is good to keep simple, always have the principle to acknowledge people for their contributions.

Please go through the following instructions before submitting the report

#### **Academic Integrity**

Academic integrity — scholarship free of fraud and deception — is an important educational objective of Penn State. Academic dishonesty can lead to a failing grade or referral to the [Office of Student Conduct](http://www.sa.psu.edu/ja/).

Academic dishonesty includes, but is not limited to:

* cheating
* plagiarism
* fabrication of information or citations
* facilitating acts of academic dishonesty by others
* unauthorized prior possession of examinations
* submitting the work of another person or work previously used without informing the instructor and securing written approval
* tampering with the academic work of other students

#### How Academic Integrity Violations Are Handled

In cases where academic integrity is questioned, [procedure requires an instructor to notify a student](http://www.psu.edu/oue/aappm/G-9-academic-integrity.html) of suspected dishonesty before filing a charge and recommended sanction with the college. Procedures allow a student to accept or contest a charge. If a student chooses to contest a charge, the case will then be managed by the respective college or campus Academic Integrity Committee. If a disciplinary sanction also is recommended, the case will be referred to the [Office of Student Conduct](http://www.sa.psu.edu/ja/title=).

All Penn State colleges abide by this Penn State policy, but review procedures may vary by college when academic dishonesty is suspected. Information about Penn State's academic integrity policy and college review procedures is included in the information that students receive upon enrolling in a course.

Additionally, Penn State students are expected to act with civility and personal integrity; respect other students' dignity, rights, and property; and help create and maintain an environment in which all can succeed through the fruits of their own efforts. An environment of academic integrity is requisite to respect for oneself and others, and a civil community.

#### For More Information on Academic Integrity at Penn State

Please see the [Academic Integrity Chart](http://www.campuses.psu.edu/CAO.pdf)  for specific college contact information or visit one of the following URLs:

* Penn State Senate [Policy on Academic Integrity](http://www.psu.edu/dept/oue/aappm/G-9.html)
* [iStudy for Success!](http://istudy.psu.edu/tutorials/) — learn about plagiarism, copyright, and academic integrity through an educational module
* [Turnitin](http://tlt.its.psu.edu/turnitin) a web-based plagiarism detection and prevention system

# Week 2 Predictive / Descriptive Analytics System Group-Based Assignment

**Abstract:**

In the modern era, people live with many of products that are provided by other instances. And in this digitalized era, where information exchange process becomes much easier than before, anyone can receive and give any information from any source. Many businesses, in the way to announce the presence of their products, compete to each other by creating compelling and captivating advertisement in order to gain some prospective customers. What exactly is advertising? It is defined as a form of communication in which a product, brand, or service is advertised to a target audience in order to generate interest, engagement, and sales. Advertisements occur in a variety of forms, ranging from text to interactive video, and have evolved to become an important element of the app marketplace. The issue arises when some advertisings are not efficient enough in reaching their intended audience. Some advertisings are simply uninteresting, and people will disregard them regardless of whether the product is good or not.

What determines if an advertisement is excellent or bad? How do you create effective advertising that will bring in new customers for your business? We will attempt to address these issues by analyzing the characteristics of good and bad advertising. For example, what song is used in an advertisement, how many individuals are featured in it, how many seconds are spent showcasing the real product, where is the logo placed, and many other elements that can be considered in the analysis. We have a dataset from a marketing research company that has all of the attributes that would support our analysis. The purpose of this project is to help any businesses to advertise their product by focusing on some attributes that can help them advertise their products effectively.

**Business Objective:**

Throughout this project, we are trying to answer these questions:

* Can we classify good and bad advertisements based on the data that we have?
* What is the smallest set of variables that need to be changed to maximize likelihood of the advertisement being categorized as good?
* Can we recommend which set of attributes is the best based on advertisement’s category?
* Which form of advertisement format that has higher impact than other?

We believe that by answering these questions, we will be able to reach the main purpose of this project, that is creating an effective advertising strategy for any kind of business who want to make a good advertisement for their product.

**Tasks:**

The next tasks that we are going to do are:

**Data Collecting**

We already have the main dataset that we are going to use for the analysis. The dataset itself contains many attributes of good and bad advertisements. If some additional dataset is needed to help the analysis, we will start to plan to collect relevant data that can refine the main dataset.

**Data Preprocessing**

The main dataset that we have contains lot of attributes. Some of them are relevant, some of them might be irrelevant to the analysis. Some of the columns also contain missing data. We need to preprocess it in order to enhance the accuracy of the analysis.

**Data Cleaning**

We will clean the data by modifying the data so that they are consistent, stripping the data values by removing the spaces, handling null values and missing values.

**Choosing a Model**

We have classification problem, so in this we will decide on model.

We will update the steps as and when we decide to move forward

# WEEK 4 Predictive/Descriptive Analytics System Group-Based Assignment

**DATASET**

We will be using dataset which is collected from MetrixLab (Market Research Company). The data consists of Advertisement attributes from various individual brands. The name of the dataset is *UL 2022 GLOBAL META DATA FILE.xslx.* The data is collected from 2019 to 2022.

Table

Description automatically generated

Figure 1: Snapshot of the Dataset

The dataset consists of total 454 Columns. With 271 Columns being the attributes of the advertisements. There is also a column which consists of the classification of the advertisement. The advertisements are classified as GOOD+ and NG+. We will use this column as our target variable for the classifying model.

The attribute columns in the dataset mainly consist of Binary(Yes/No) data. Based on this observation, we are pretty sure that there is no outlier in our dataset. There are some missing values found in the dataset. We have to clean this dataset in our next step.

Unfortunately, there is no way that we can collect data from other source that is compatible with our main dataset. As our main dataset contains specific attributes for the advertisements that can’t be matched with other data.

Chart, bar chart

Description automatically generated

Figure 2: Count Based on Categories

Based on the count, we can see that the dataset does not have a balanced dataset in terms of categories. The category that shows the most in the dataset is from Personal Care advertisement. And the category that shows the least is from Nutrition advertisement.

Chart, bar chart

Description automatically generated

Figure 3: Count Based on Classification

Based on the classification count, we can see that again, the dataset does not have balanced dataset with the classification. The advertisements that are classified as bad are more in count compared to the advertisements that are classified as good. We can see from the counts that are related to our business questions, there is data bias in our dataset. We have to reduce this bias in our next steps.

**Tasks:**

The next things that we are going to do towards the project:

**Data Preparation**

Our next goal is to list all the data preparation steps that we need to do for this project.

**Data Cleaning**

The data that we have is still not clean enough for the analysis. There are missing values found in the data. And there are data bias that is present in the data. We will have to figure out how to handle these problems.

# WEEK 6 Predictive Analytics System Group-Based Assignment

**DATA PREPARATION PLAN**

* **Data Structuring**
  + Extracting the data from original dataset into separated CSV files

In the original dataset that we have, the data contains some other non-relevant sheets and the data itself is in excel format. To maintain convenience, we plan to restructure the data by extracting the part of the dataset that we are interested in and changing the format to CSV file.

* **Data Cleaning**
  + Handling missing values by filling them

The dataset contains missing values in some of the columns. We are planning to handle the missing values by filling them. We are not going to drop the observations that have missing values because our total observations are not large enough. If we drop the missing values, we will be left with only few of total observations.

* + Dropping unwanted characters, symbols, and whitespaces from categorical values

The dataset that we have is not clean enough for the analysis. There are some unwanted characters and symbols that can interrupt with the analysis process. We are planning to remove these all.

* + Checking outliers in the data for numerical values

We are planning to check the columns of numerical values for outliers that might present in the dataset. If we found any outlier, we will change it into mean value of the column.

* + Removing irrelevant observations

If there’s any irrelevant observation that presents in the dataset, we will remove them immediately as it won’t be giving any usefulness to the analysis.

* **Data Transformation**
  + Data Scaling/Normalization

Normalization is a technique often applied as part of data preparation for machine learning. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information. We are going to normalize the numeric values of the dataset so that every value is in the same scale.

* + Lower-casing text

To maintain uniformity of the values, we plan to transform all categorical values in the dataset to lower case. This step helps to prevent high cardinality in our data.

* + Changing binary (yes/no) to binary (1/0)

The binary columns that we have in our dataset is in yes/no values. We plan to change these values to 1/0. This step helps the analysis process because it is easier to process the 1/0 value.

* **Dimensionality Reduction**
  + Selecting feature that holds relevancy to the analysis

Feature selection is a way of selecting the subset of the most relevant features from the original features set by removing the redundant, irrelevant, or noisy features. While developing the machine learning model, only a few variables in the dataset are useful for building the model, and the rest features are either redundant or irrelevant. If we input the dataset with all these redundant and irrelevant features, it may negatively impact and reduce the overall performance and accuracy of the model. Hence it is very important to identify and select the most appropriate features from the data and remove the irrelevant or less important features. Our data has so many columns. We plan to select features that is relevant to our analysis so that we maintain the best performance to the model

* + Factor Analysis

Factor Analytics is a special technique reducing the huge number of variables into a few numbers of factors is known as factoring of the data, and managing which data is to be present in sheet comes under factor analysis. The factor analysis technique extracts the maximum common variance from all the variables and puts them into a common score. We plan to use this technique to further reduce the features to reasonable number.

* + Filtering highly correlated features

The goal of filtering highly correlated features is to prevent the features to decrease the performance of the model. As the highly correlated value won’t give anything valuable to the analysis, we will filter it out if there is any of it in our dataset.

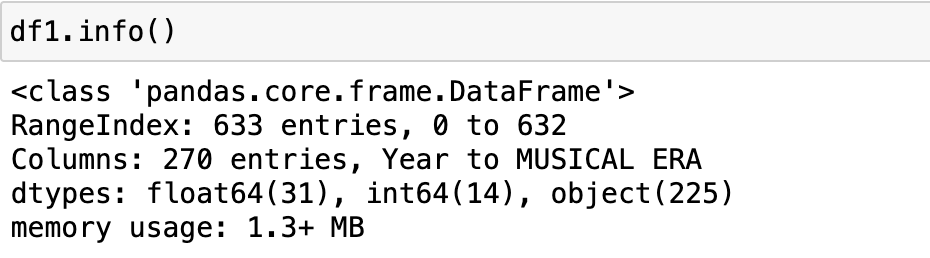
# 

# WEEK 8 Predictive/descriptive Analytics System Group-Based Assignment

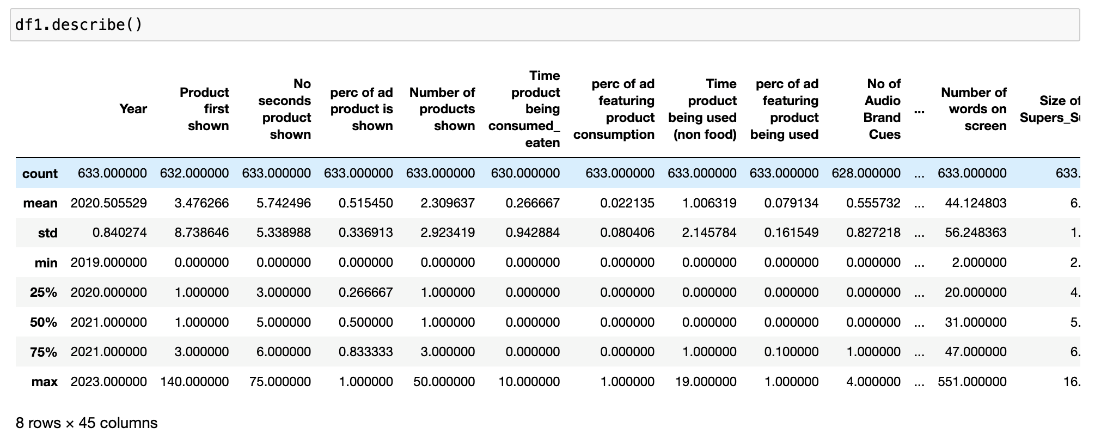
**Predictive/Descriptive Analysis Plan :**

Exploratory Data Analysis:

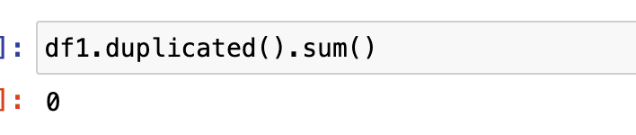
When we see the info of data. We can see we have270 columns.



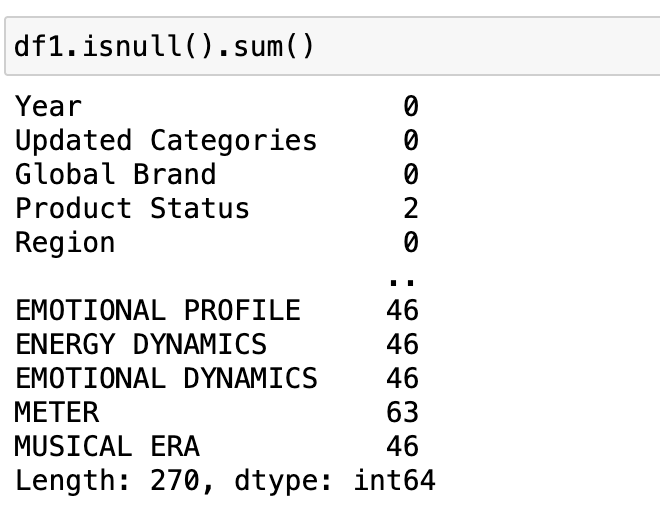
Describe data as well shows 45 columns which tells us, 45 columns are of type number and remaining all belongs to cat4egorical data.



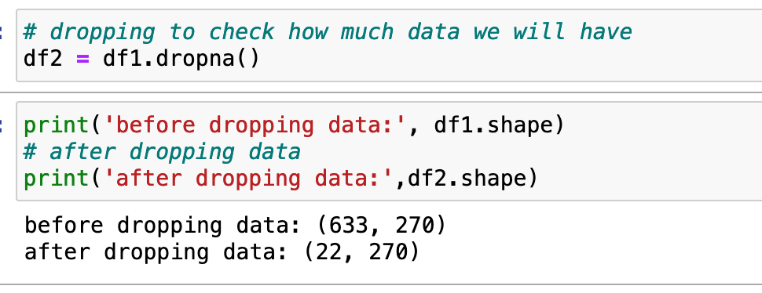
There are no duplicates present in the data.



We see mostly null data on categorical data,



We tried to remove Null values to check how our data would look once all the null is removed.

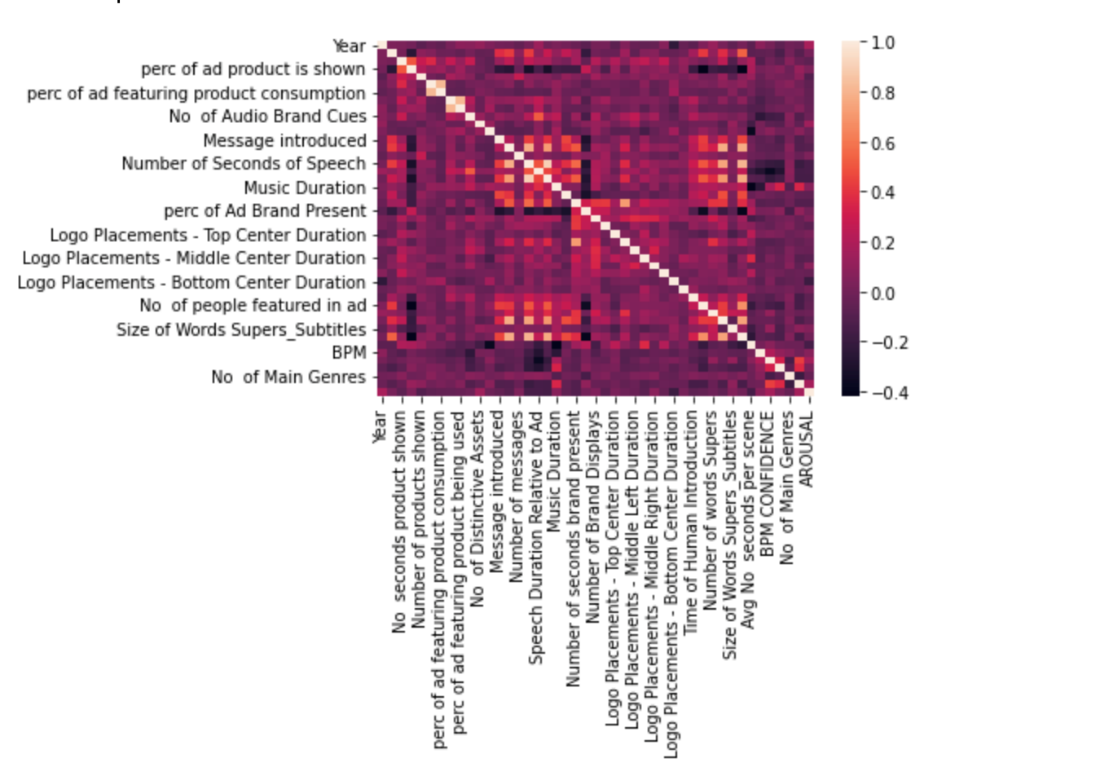


We will end up having only 22 data points, which would be way to less to proceed with making a model.

This overview would say that we have data with most of it being missed.

Hence Data imputation would be critical for us in this case.

Just to show the correlation matrix before dropping features which are not making impact.



This correlation makes no sense without dimensionally reduction/feature extraction.

For our next step would be performing, data imputation and elimination of unimportant features.

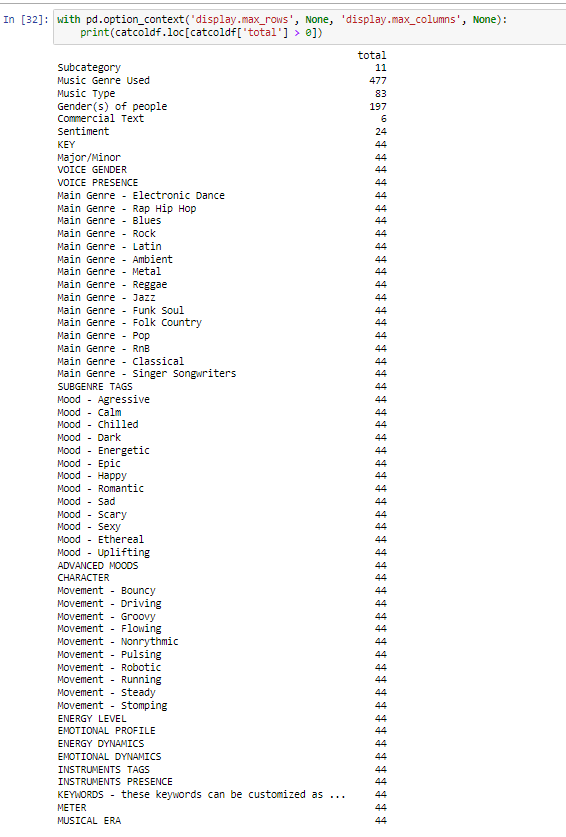
**Data Imputation:**

Missing values are filled with mean values for numerical data, for categorical data, we plan to impute those with most frequent class.

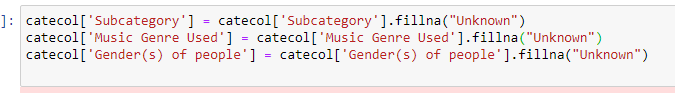
We first have to split the columns that contain numerical and categorical data as the treatment of both numeric and categorical data will be different from each other. We store both of numerical and categorical data in different data frames for easier management.

Categorical data:

We first have to check which columns have the missing values. And we also have to check how many missing rows are contained in the columns. Below are the columns that have missing values in the observation rows:



As we can see, there are some columns that have high numbers of missing values. Filling them with most occurring values won’t be appropriate to the data. So, we just simply fill them with “Unknown”.

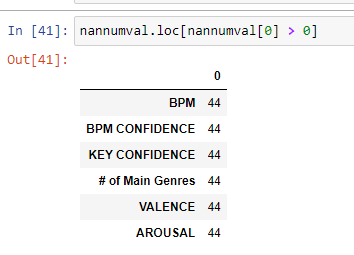


For remaining columns that have missing values, we input them with the most occurring values in the column.

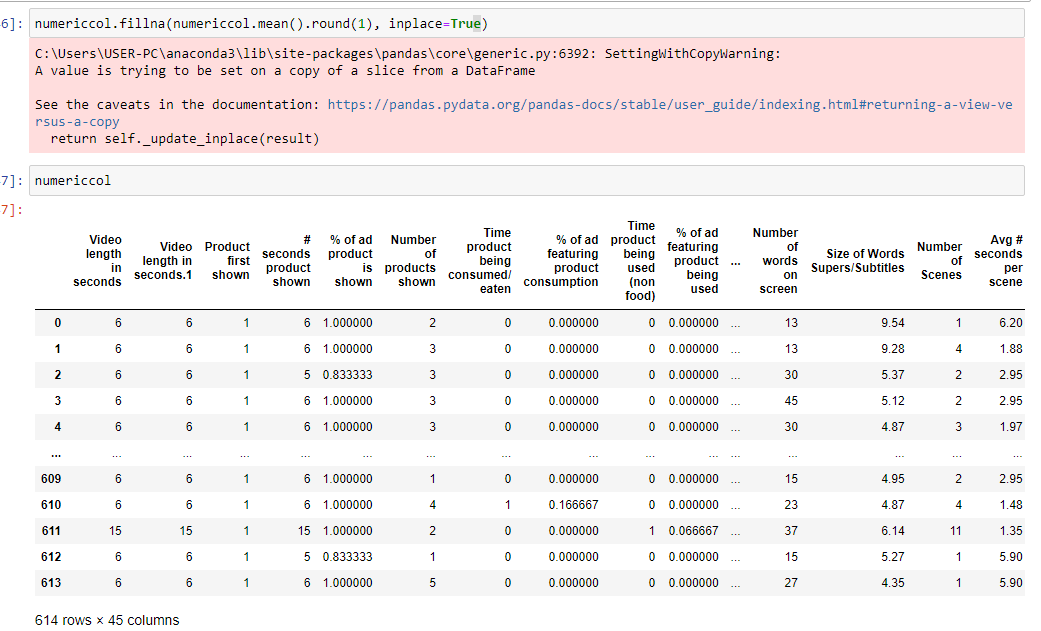


Numerical data:

For numerical data, we also have to check which columns have missing values and we count how many missing values to determine our next steps to treat them. Below are the columns that have missing values in the numerical section:

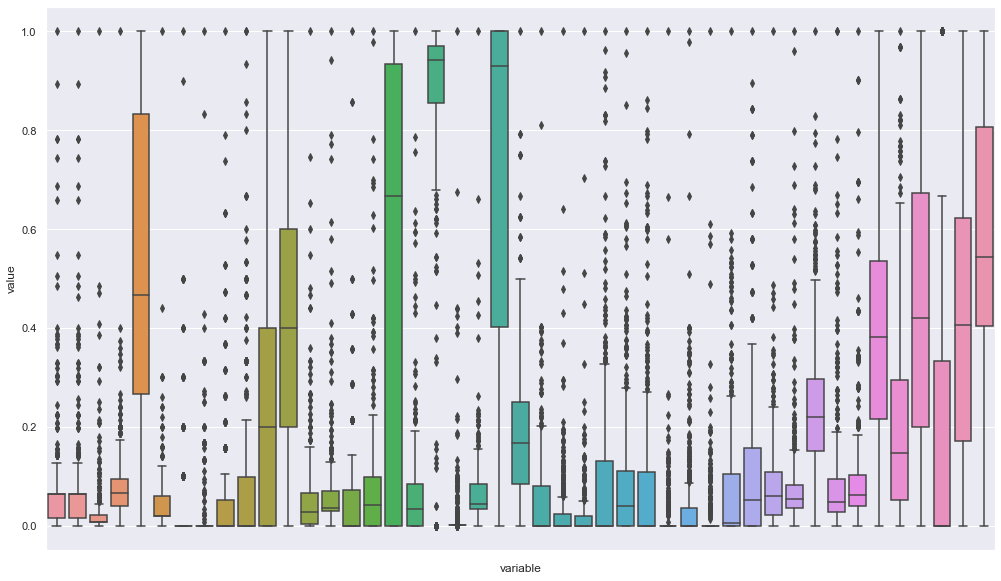


As we can see, the missing rows on the numerical data are not so many. We can easily handle this by inputting mean values of the respecting columns.



**Numerical Outliers:**

We tried to check if our data contains outliers or not. We created boxplots to see if the numerical columns in our dataset have outliers. Below are the results:



As we can see, there are lots of values that are located beyond the min and max whiskers of the plots. We will transform these values into mean value of the columns without the outliers. Detailed steps will be explained later.

**Encoding:**

We plan to perform the encoding for categorical data as below and take a call based on the outcome of the result.

**One Hot Encoding:**

One hot encoding is a process of converting categorical data variables so they can be provided to machine learning algorithms to improve predictions. One hot encoding is a crucial part of feature engineering for machine learning.

With one-hot, we convert each categorical value into a new categorical column and assign a binary value of 1 or 0 to those columns. Each integer value is represented as a binary vector. All the values are zero, and the index is marked with a 1.

Once we assign numeric values, we create a binary vector that represents our numerical values.

We can use **get\_dummies** to perform one-hot encoding.

Since our data has lot of features, this step again may produce larger data set.

So, we will also proceed with performing a binary encoding to check which one of these 2 can be considered for encoding.

**Binary Encoding:**

Binary encoding is a process where we can perform hash encoding look like encoding without losing the information just like one hot encoding.

We can say that binary encoding is a combination process of hash and one hot encoding.

We can see without losing much information we can get encoded data with reduced dimensionality than the One-Hot encoding in binary encoding. This encoding is very helpful in the case of data with a huge number of categories like our dataset.

**Feature Extraction:**

We will perform the below feature extraction and based on the result we will finalize the feature extraction method for our data set.

**Factor Analysis:**

We will perform this using the python factor\_analyzer library.

The primary objective of factor analysis is to reduce the number of observed variables and find unobservable variables.

**Correlation :**

Correlation is a statistical term which in common usage refers to how close two variables are to having a linear relationship with each other.

Features with high correlation are more linearly dependent and hence have almost the same effect on the dependent variable. So, when two features have high correlation, we can drop one of the two features.

Removal of different features from the dataset will have different effects on the p-value for the dataset. We can remove different features and measure the p-value in each case. These measured p-values can be used to decide whether to keep a feature or not.

**RFE :**

If the above feature extraction doesn’t give satisfying results, we would perform RFE.

RFE works by searching for a subset of features by starting with all features in the training dataset and successfully removing features until the desired number remains.

This is achieved by fitting the given machine learning algorithm used in the core of the model, ranking features by importance, discarding the least important features, and re-fitting the model. This process is repeated until a specified number of features remains.

# WEEK 10 Predictive/descriptive Analytics System Group-Based Assignment

**DATA TRANSFORMATION:**

**Lower casing text values**

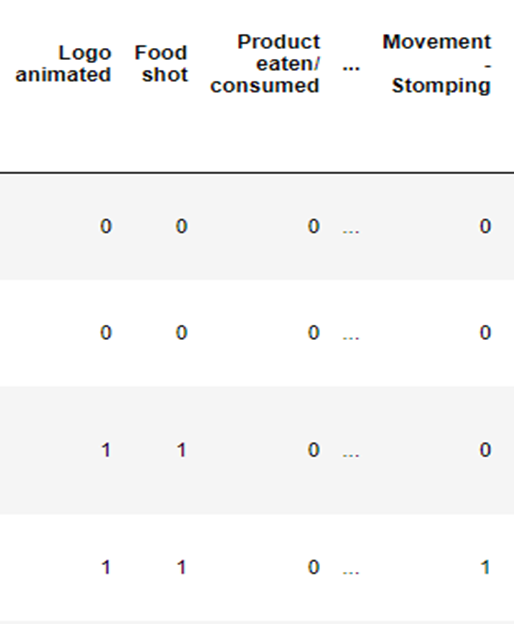
The first transformation that we did is lower casing all categorical value in the data. To maintain uniformity of the values, we transform all categorical values in the dataset to lower case. This step helps to prevent high cardinality in our data. The result of the lower case is as following.

Graphical user interface, text

Description automatically generated

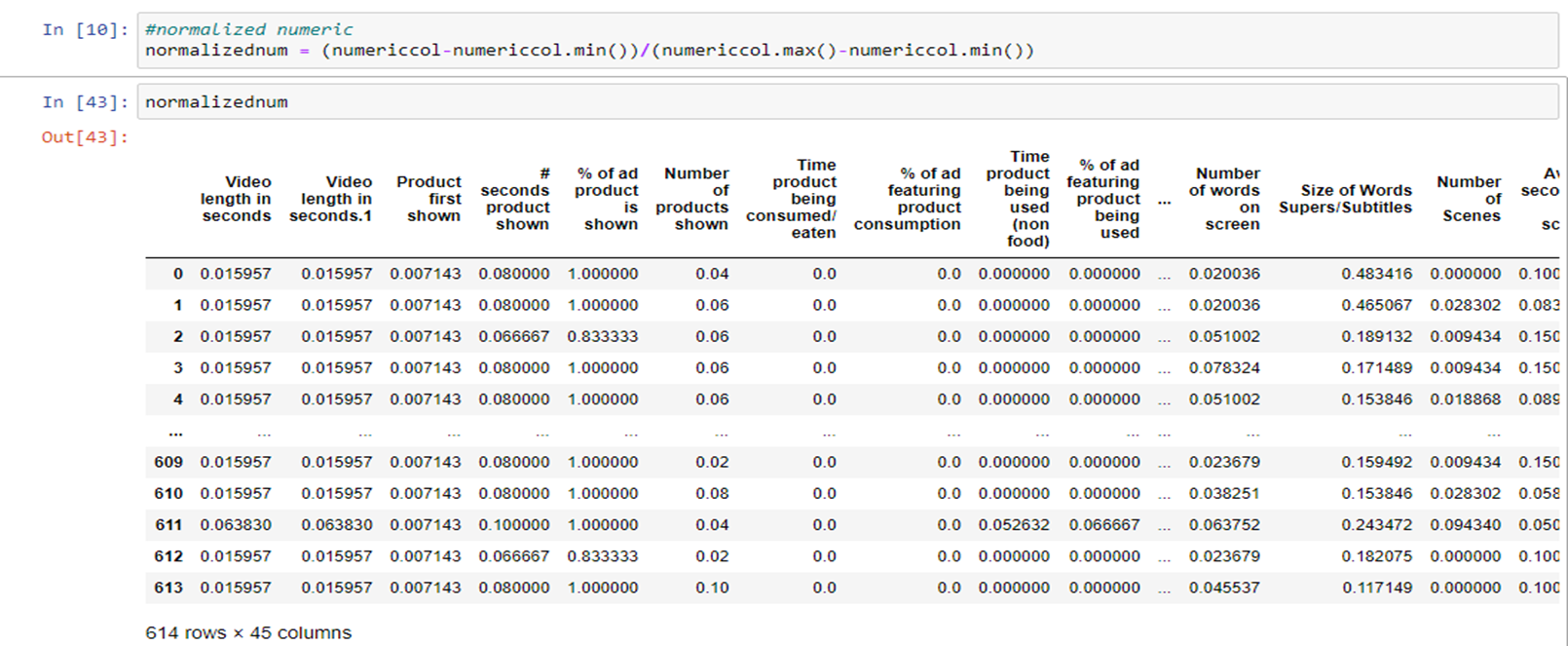
**Changing yes/no binary data to 1/0**

Next transformation that we did is changing binary categorical to 1/0 data. This step helps the analysis process because it is easier to process the 1/0 value. And obviously we only can process numeric data in the analysis. The result after transformation is as following.



**Normalizing numeric values**

Next transformation step that we did was normalizing numeric values that are available in our data. The goal of normalization is to change the values of numeric columns in the dataset to use a common scale, without distorting differences in the ranges of values or losing information. The result is as following.



**Changing categorical values to ordinal encoding**

The next transormation step that we did is encode categorical values with ordinal encoding. ordinal encoding converts each label into integer values and the encoded data represents the sequence of labels. The reason of this step is to change categorical values to some format so that it can be analyzed in the modelling step. The result is as following.

Table

Description automatically generated

**Changing outliers to mean value without outliers**

As we mentioned before, we plan to change outliers in numeric to the mean value after taking out the outlier. This step helps to transform outliers that are present in the data to some kind of soft outliers that don’t interrupt with the analysis. The boxplot after this step is as following.

Chart

Description automatically generated

**Data Upsampling**

Upsampling is a procedure where synthetically generated data points (corresponding to minority class) are injected into the dataset. After this process, the counts of both labels are almost the same. This equalization procedure prevents the model from inclining towards the majority class.  We did upsampling to balance the data because the NG classified ads are the most common in the dataset. Here is the plot of before and after data upsampling.

Before:

**Chart, pie chart

Description automatically generated**

After:

**Chart, pie chart

Description automatically generated**

**Label Encoding**

**Label Encoding** refers to converting the labels into a numeric form so as to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated. It is an important pre-processing step for the structured dataset in supervised learning.

This type of encoding is used when the variables in the data are ordinal, ordinal encoding converts each label into integer values and the encoded data represents the sequence of labels.

We decided to use label encoding because if we use the one hot encoding, the number of columns will increase in response of the encoding itself. Here is the result after applying label encoding.

**A picture containing table

Description automatically generated**

**Correlation of features**

In order to eliminate correlated features and to reduce the dimension in the dataset, we tried to see if there are highly correlated features that pass the threshold that we set. Here are the results.

Table

Description automatically generated

Some columns passed the threshold. We can see that there are columns that are highly correlated with each other. We immediately remove the columns to reduce the number of features.

**Boruta**

Boruta is a Machine Learning algorithm used in feature selection. Feature selection is a process of reducing the number of features in a dataset by identifying features that largely influence the study variable.

It is an important aspect of machine learning, for instance, supposing we are required to perform an analysis of genetic data. The dataset, in this case, can be very huge, and fitting the machine learning model in it may have significant challenges.

Boruta is a wrapper method of the Feature selection built around the Random Forest Classifier algorithm. The algorithm works by taking features of the original dataset and creates a copy of them. On this copy, values in each column are shuffled to attain randomness.

These shuffled features are known as Shadow Features. The shadow features are then merged with the original features to obtain a new feature space whose dimension is twice the original dataset. The diagram below clarifies the above discussion.

We tried Boruta on our data to see optimal features that we can use to the classifier. Here are the results.

Table

Description automatically generated

Graphical user interface, text, application

Description automatically generated

We can see that the algorithm confirmed 51 features, hold 7 features on tentative, rejected 146 features on 100 iterations. We can use these features that passed the Boruta to the classifier.

Things that we are planning to do after these steps are:

* Standardization of numeric data
* Once these done, we can focus on removing correlated data/unwanted columns.

# WEEK 12 Predictive / Descriptive Analytics System Group-Based Assignment

**DATA MODELLING:**

***Evaluation Metrics Used***

**Accuracy**

Accuracy is the most intuitive performance measure, and it is simply a ratio of correctly predicted observation to the total observations. Accuracy may not be a good measure if the dataset is not balanced (both negative and positive classes have different number of data instances).

**Precision**

Precision is the ratio of correctly predicted positive observations to the total predicted positive observations. Precision should ideally be 1 (high) for a good classifier. Precision becomes 1 only when the numerator and denominator are equal i.e TP = TP +FP, this also means FP is zero. As FP increases the value of denominator becomes greater than the numerator and precision value decreases.

**Recall**

Recall is the ratio of correctly predicted positive observations to all observations in actual class. Recall should ideally be 1 (high) for a good classifier. Recall becomes 1 only when the numerator and denominator are equal i.e TP = TP +FN, this also means FN is zero. As FN increases the value of denominator becomes greater than the numerator and recall value decreases.

**F1 Score**

F1 Score is the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account. Intuitively it is not as easy to understand as accuracy, but F1 is usually more useful than accuracy, especially if you have an uneven class distribution.

**Confusion Matrix**

Confusion matrix, also known as an error matrix, is a specific table layout that allows visualization of the performance of an algorithm, typically a supervised learning one (in unsupervised learning it is usually called a matching matrix). Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa – both variants are found in the literature. The name stems from the fact that it makes it easy to see whether the system is confusing two classes (i.e. commonly mislabeling one as another).

It is a special kind of contingency table, with two dimensions ("actual" and "predicted"), and identical sets of "classes" in both dimensions (each combination of dimension and class is a variable in the contingency table).

***Classifiers used and results***

**Random Forest Classifier**

Random forest is a flexible, easy-to-use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most-used algorithms, due to its simplicity and diversity. It can be used for both classification and regression tasks.

Random forest is a supervised learning algorithm. The “forest” it builds is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

We did some classifying random forest to test our data. We did Random Forest because we want to use ensemble technique to the data. The result is as following.

Chart, treemap chart

Description automatically generated

The errors are mainly on false positive side.

Accuracy:  99%

F1 score: 99%

Recall: 99%

Precision: 99%

We also look into the feature importance rank of random forest model to see which features being the most impactful in the classifying process. Here is the result.

A picture containing shape

Description automatically generated

We can see that top 5 features that being the most impactful based on random forest classifier are Noticability threshold, Number of Scenes, Country, Number of Distinctive assets, percentage of ad product shown.

**Gradient Boosting Classifier**

Gradient boosting is a type of machine learning boosting. It relies on the intuition that the best possible next model, when combined with previous models, minimizes the overall prediction error. The key idea is to set the target outcomes for this next model in order to minimize the error. The target outcome for each case in the data depends on how much changing that case's prediction impacts the overall prediction error:

* If a small change in the prediction for a case causes a large drop in error, then next target outcome of the case is a high value. Predictions from the new model that are close to its targets will reduce the error.
* If a small change in the prediction for a case causes no change in error, then next target outcome of the case is zero. Changing this prediction does not decrease the error.

The name gradient boosting arises because target outcomes for each case are set based on the gradient of the error with respect to the prediction. Each new model takes a step in the direction that minimizes prediction error, in the space of possible predictions for each training case.

We chose Gradient Boosting to see the effect of minimized bias error to the data. Here are the results after applying gradient boosting to our data.

Chart, treemap chart

Description automatically generated

The errors are mainly on false positive side.

Accuracy: 99%

F1 score: 99%

Recall: 99%

Precision: 99%

We also look into the feature importance rank of gradient boosting model to see which features being the most impactful in the classifying process. Here is the result.

Shape

Description automatically generated

We can see that top 5 features that being the most impactful based on gradient boosting classifier are Noticability threshold, duration of logo placement in middle right side, number of seconds of speech, logo in ending video, percentage of ad product shown.

**Future Improvement**

We are only doing the modelling on the categorical features. For next improvement, we will try to apply the numeric features of the dataset. And add other machine learning algorithm to the data. Like Logistic Regression, Support Vector Machine, and Naïve Bayes. And also we will try to do hyperparameter tuning to get the best set of settings for the classifier.

**Testing Training Strategy**

The train-test split procedure is used to estimate the performance of machine learning algorithms when they are used to make predictions on data not used to train the model. It is a fast and easy procedure to perform, the results of which allow you to compare the performance of machine learning algorithms for your predictive modeling problem. Although simple to use and interpret, there are times when the procedure should not be used, such as when you have a small dataset and situations where additional configuration is required, such as when it is used for classification and the dataset is not balanced.

For the data split strategy, we use the 80/20 ratio. 80% for training data. And 20% for test data. This split is widely used in machine learning world because it follows pareto principle. This split is also good because the 80% ratio tend to prevent the classifier to overfit the data.

**Validation**

Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample.

The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation.

Cross-validation is primarily used in applied machine learning to estimate the skill of a machine learning model on unseen data. That is, to use a limited sample in order to estimate how the model is expected to perform in general when used to make predictions on data not used during the training of the model.

For validation, we use K-Fold Cross Validation Strategy. With 10-fold as the split number. The objective is we want to make sure which of the algorithms suit the data best. Following is the result of the validation.

Text

Description automatically generated

We can see that Gradient Boosting (GR), Random Forest (LDA), and AdaBoost (ADA) give the best performance on the data that we have.

**Hyper Parameter Tuning**

Hyperparameter tuning consists of finding a set of optimal hyperparameter values for a learning algorithm while applying this optimized algorithm to any data set. That combination of hyperparameters maximizes the model’s performance, minimizing a predefined loss function to produce better results with fewer errors.

For getting the best parameter, we use RandomizedSearchCV. Which tries random combinations of a range of values (we have to define the number of iterations). It is good at testing a wide range of values and normally it reaches a very good combination very fast, but the problem that it doesn’t guarantee to give the best parameter combination because not all parameter values are tried out. The strategy is the same as algorithm validation. With 10-fold as the split number. Following is the result of the best parameters.

Random Forest:

'random\_state': 42, 'n\_estimators': 100, 'min\_samples\_split': 10, 'min\_samples\_leaf': 11, 'max\_depth': 6, 'bootstrap': True

Gradient Boosting

'n\_estimators': 50, 'max\_depth': 7, 'learning\_rate': 1

AdaBoost

'n\_estimators': 50, 'learning\_rate': 0.01

Logistic Regression

'solver': 'newton-cg', 'penalty': 'l2', 'C': 100

Support Vector Machine

'kernel': 'poly', 'gamma': 0.001, 'C': 1

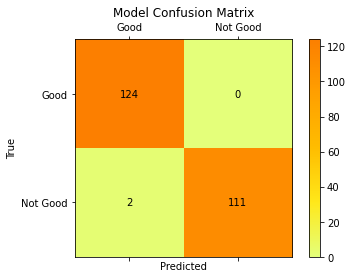
**Additional Algorithms to be applied**

AdaBoost

AdaBoost also called Adaptive Boosting is a technique in Machine Learning used as an Ensemble Method. The most common algorithm used with AdaBoost is decision trees with one level that means with Decision trees with only 1 split. These trees are also called Decision Stumps.

What this algorithm does is that it builds a model and gives equal weights to all the data points. It then assigns higher weights to points that are wrongly classified. Now all the points which have higher weights are given more importance in the next model. It will keep training models until and unless a lower error is received.

We chose to apply this algorithm because we want to see if another ensemble technique other than random forest can yield different result to the data. Following is the result for AdaBoost.



We can see that the model did a good job. And the error is in the false positive side.

Accuracy: 99%

F1 score: 99%

Recall: 99%

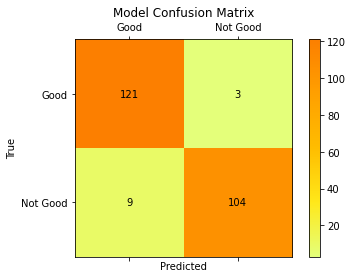
Precision: 99%

Support Vector Machine

SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the data into classes. The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space (N — the number of features) that distinctly classifies the data points.

Hyperplanes are decision boundaries that help classify the data points. Data points falling on either side of the hyperplane can be attributed to different classes. Also, the dimension of the hyperplane depends upon the number of features. If the number of input features is 2, then the hyperplane is just a line. If the number of input features is 3, then the hyperplane becomes a two-dimensional plane. It becomes difficult to imagine when the number of features exceeds 3.

We chose this algorithm because SVM tend to work best on high dimensional data. And our data consists of many dimensions. Following is the result for SVM.



We can see that the error is mainly on the false positive side.

Accuracy: 95%

F1 score: 95%

Recall: 95%

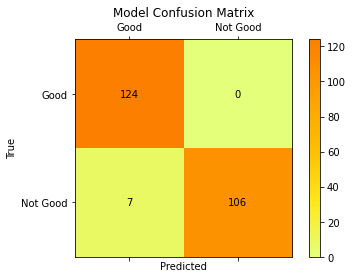
Precision: 95%

Logistic Regression

Logistic regression is a statistical analysis method to predict a binary outcome, such as yes or no, based on prior observations of a data set. A logistic regression model predicts a dependent data variable by analyzing the relationship between one or more existing independent variables.

Logistic regression has become an important tool in the discipline of machine learning. It allows algorithms used in machine learning applications to classify incoming data based on historical data. As additional relevant data comes in, the algorithms get better at predicting classifications within data sets. Logistic regression can also play a role in data preparation activities by allowing data sets to be put into specifically predefined buckets during the extract, transform, load (ETL) process in order to stage the information for analysis.

We chose this algorithm because the data that we use consists of binary classification. That is not good and good. Following is the result of Logistic Regression.



We can see that the error is only on the false positive side.

Accuracy: 97%

F1 score: 97%

Recall: 97%

Precision: 97%

Naïve bayes

Naïve bayes is a classification technique based on Bayes’ Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Naive Bayes is a probabilistic algorithm that’s typically used for classification problems. Naive Bayes is simple, intuitive, and yet performs surprisingly well in many cases.

Naïve Bayes is a probabilistic machine learning algorithm based on the Bayes Theorem, used in a wide variety of classification tasks. Bayes’ Theorem is a simple mathematical formula used for calculating conditional probabilities. Conditional probability is a measure of the probability of an event occurring given that another event has (by assumption, presumption, assertion, or evidence) occurred.

We chose this algorithm to see the effect of applying probabilistic Bayes Theorem to our data. Following is the result of Naïve bayes.

Chart, treemap chart

Description automatically generated

We can see that the error is same on false positive and false negative.

Accuracy: 98%

F1 score: 98%

Recall: 98%

Precision: 98%

**Advertisement Format Recommendation**

Based on the samples that we get from the overall good classification; we plot the type of ad formatting and the count of it. Here is the result.

Chart, bar chart

Description automatically generated

We can see from the plot, that online video has been the most effective format in advertisement. Followed by social media video and generic video. We can see that advertising in Facebook video is not recommended because it yield the least count in good advertisement.

# References:

Hoare, J. (2022, August 24). *Gradient boosting explained - the coolest kid on the Machine Learning Block*. Displayr. Retrieved November 29, 2022, from https://www.displayr.com/gradient-boosting-the-coolest-kid-on-the-machine-learning-block/

Morde, V. (2019, April 8). *XGBoost algorithm: Long may she reign!* Medium. Retrieved November 29, 2022, from https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d

*How to get started with the Boruta algorithm in Machine Learning*. Section. (n.d.). Retrieved November 29, 2022, from https://www.section.io/engineering-education/getting-started-with-boruta-algorithm/

B, H. N. (2020, June 1). *Confusion matrix, accuracy, precision, recall, F1 score*. Medium. Retrieved November 29, 2022, from https://medium.com/analytics-vidhya/confusion-matrix-accuracy-precision-recall-f1-score-ade299cf63cd

Wikimedia Foundation. (2022, August 31). *Confusion matrix*. Wikipedia. Retrieved November 29, 2022, from https://en.wikipedia.org/wiki/Confusion\_matrix

Brownlee, J. (2020, August 26). *Train-test split for Evaluating Machine Learning Algorithms*. MachineLearningMastery.com. Retrieved November 29, 2022, from https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/

Brownlee, J. (2020, August 2). *A gentle introduction to k-fold cross-validation*. MachineLearningMastery.com. Retrieved November 29, 2022, from https://machinelearningmastery.com/k-fold-cross-validation/

Saini, A. (2021, September 15). *AdaBoost algorithm - A complete guide for beginners*. Analytics Vidhya. Retrieved November 29, 2022, from https://www.analyticsvidhya.com/blog/2021/09/adaboost-algorithm-a-complete-guide-for-beginners/

Pupale, R. (2019, February 11). *Support vector machines(svm) - an overview*. Medium. Retrieved November 29, 2022, from https://towardsdatascience.com/https-medium-com-pupalerushikesh-svm-f4b42800e989

Gandhi, R. (2018, July 5). *Support Vector Machine - introduction to machine learning algorithms*. Medium. Retrieved November 29, 2022, from https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

Lawton, G., Burns, E., & Rosencrance, L. (2022, January 20). *What is logistic regression? - definition from Searchbusinessanalytics*. SearchBusinessAnalytics. Retrieved November 29, 2022, from https://www.techtarget.com/searchbusinessanalytics/definition/logistic-regression#:~:text=Logistic%20regression%20is%20a%20statistical,or%20more%20existing%20independent%20variables.

*Naïve Bayes Algorithm: Everything you need to know*. KDnuggets. (n.d.). Retrieved November 29, 2022, from https://www.kdnuggets.com/2020/06/naive-bayes-algorithm-everything.html

Ray, S. (2022, November 29). *Learn naive Bayes algorithm: Naive Bayes classifier examples*. Analytics Vidhya. Retrieved November 29, 2022, from https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/#:~:text=Naive%20Bayes%20Model-,What%20is%20Naive%20Bayes%20algorithm%3F,presence%20of%20any%20other%20feature.

**APPENDIX**

**Code Script:**

**#!/usr/bin/env python**

**# coding: utf-8**

**# In[1]:**

**import time**

**import pandas as pd**

**import numpy as np**

**import os**

**from pathlib import Path**

**import warnings**

**warnings.filterwarnings("ignore")**

**from sklearn.preprocessing import StandardScaler**

**# standardizing data**

**from sklearn.preprocessing import LabelEncoder, OrdinalEncoder**

**from sklearn.model\_selection import cross\_val\_score**

**from sklearn.metrics import confusion\_matrix, accuracy\_score, precision\_score, recall\_score, f1\_score**

**from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, AdaBoostClassifier**

**from sklearn.linear\_model import LogisticRegression, LinearRegression**

**from sklearn.naive\_bayes import GaussianNB**

**from sklearn.neighbors import KNeighborsClassifier**

**from sklearn.svm import SVC**

**#import xgboost as xgb**

**import matplotlib.pyplot as plt**

**from sklearn.model\_selection import train\_test\_split**

**# In[ ]:**

**#conda install py-xgboost**

**# In[2]:**

**df1 = pd.read\_excel('Savedata.xlsx',skiprows=1)**

**# In[3]:**

**df1 = df1.iloc[:,1:]**

**# In[4]:**

**# dropping columns irrelevant**

**df1 = df1.drop(['VALENCE', 'AROUSAL'], axis=1)**

**# In[5]:**

**# removing whitespaces**

**df1[df1.columns] = df1[df1.columns].apply(lambda x: x.str.strip() if (x.dtype == 'object') else x)**

**# check for special character in the columns**

**# In[6]:**

**df1.columns**

**# In[7]:**

**import janitor as jn**

**df1 = jn.clean\_names(df1)**

**df1**

**# In[8]:**

**# printing all features**

**features = []**

**for i in df1.columns:**

**features.append(i)**

**# In[9]:**

**features.sort()**

**# In[14]:**

**for i in features:**

**print(i)**

**# In[15]:**

**df1**

**# In[16]:**

**df1.isnull().sum().to\_frame('missing\_values').reset\_index().sort\_values(by='missing\_values',ascending=False).query('missing\_values > 25')**

**# In[17]:**

**# dropping all the column which has more than 5 nulls in them**

**df1\_nan = df1.dropna(thresh=len(df1) -25, axis=1)**

**# In[18]:**

**# checking the data and data shape**

**df1\_nan**

**# In[19]:**

**print(df1\_nan.shape)**

**# In[20]:**

**import pandas as pd**

**import numpy as np**

**from sklearn.base import TransformerMixin**

**class DataFrameImputer(TransformerMixin):**

**def \_\_init\_\_(self):**

**"""Impute missing values.**

**Columns of dtype object are imputed with the most frequent value**

**in column.**

**Columns of other types are imputed with mean/median of column.**

**"""**

**def fit(self, X, y=None):**

**self.fill = pd.Series([X[c].value\_counts().index[0]**

**if X[c].dtype == np.dtype('O') else X[c].median() for c in X],**

**index=X.columns)**

**return self**

**def transform(self, X, y=None):**

**return X.fillna(self.fill)**

**# In[21]:**

**# data imputation for null data**

**df\_xt = DataFrameImputer().fit\_transform(df1\_nan)**

**# In[22]:**

**df\_xt**

**# In[23]:**

**df\_xt['classification'].unique()**

**# In[24]:**

**df\_xt.groupby(['region', 'classification'])['classification'].count()**

**# ### Combined Data (numeric and categorical)**

**# In[25]:**

**df\_good = df\_xt[df\_xt["classification"] == 'GOOD+']**

**df\_notgood = df\_xt[df\_xt["classification"] =='NG+']**

**# In[26]:**

**print(df\_xt["classification"].value\_counts())**

**df\_xt.groupby('classification').size().plot(kind='pie',**

**y = "classification",**

**label = "Type",**

**autopct='%1.1f%%')**

**# In[27]:**

**from sklearn.utils import resample**

**good\_upsample = resample(df\_good,**

**replace=True,**

**n\_samples=len(df\_notgood),**

**random\_state=42)**

**print(good\_upsample.shape)**

**# In[28]:**

**print(good\_upsample.shape)**

**print(df\_notgood.shape)**

**# In[27]:**

**# good\_upsample = good\_upsample.loc[~good\_upsample.index.duplicated(keep='first')]**

**# df\_notgood = df\_notgood.loc[~df\_notgood.index.duplicated(keep='first')]**

**# In[29]:**

**df\_notgood**

**# In[30]:**

**df\_upsampled = pd.concat([good\_upsample, df\_notgood])**

**df\_upsampled['classification'].value\_counts()**

**# In[31]:**

**df\_upsampled.shape**

**# In[31]:**

**#cols = df\_upsampled.select\_dtypes(exclude='object').columns**

**# In[32]:**

**cols = df\_upsampled.columns**

**# In[33]:**

**frames = [df\_upsampled.select\_dtypes(include='object'),df\_upsampled.select\_dtypes(exclude='object')]**

**df\_sorted = pd.concat(frames,axis=1).reset\_index(drop = True)**

**# In[34]:**

**df\_sorted**

**# In[35]:**

**df\_sorted.columns.get\_loc("avg\_no\_seconds\_per\_scene")**

**# 172-210 onwards numerical data will start and up untill 0-171 we have categorical data so will be using these to encode.**

**# In[36]:**

**cat\_col= df\_sorted.select\_dtypes(include='object').columns**

**cat\_col**

**# In[37]:**

**num\_col= df\_sorted.select\_dtypes(exclude='object').columns**

**num\_col**

**# In[38]:**

**# class ItemSelector():**

**# def \_\_init\_\_(self, key):**

**# self.key = key**

**# def fit(self, x, y=None):**

**# return self**

**# #**

**# def transform(self, data\_dict):**

**# return data\_dict[self.key]**

**# In[39]:**

**# class MyLEncoder():**

**# def transform(self, X, y=None, \*\*fit\_params):**

**# enc = preprocessing.LabelEncoder()**

**# encc = enc.fit(X)**

**# enc\_data = enc.transform(X)**

**# return enc\_data**

**# def fit\_transform(self, X, y=None, \*\*fit\_params):**

**# self.fit(X, y, \*\*fit\_params)**

**# return self.transform(X)**

**# def fit(self, X, y=None, \*\*fit\_params):**

**# return self**

**# In[40]:**

**# from sklearn.compose import ColumnTransformer**

**# from sklearn.pipeline import Pipeline**

**# from sklearn.pipeline import FeatureUnion**

**# from sklearn import preprocessing**

**# encoding\_pipeline =Pipeline([**

**# ('union', FeatureUnion(**

**# transformer\_list=[**

**# ('categorical', Pipeline([**

**# ('selector', ItemSelector(key=cat\_col)),**

**# ('LabelEncoder', MyLEncoder()) ]))**

**# ]))**

**# ])**

**# In[41]:**

**# X = df\_sorted**

**# encoding\_pipeline.fit\_transform(X)**

**# X**

**# In[42]:**

**# from sklearn.compose import ColumnTransformer**

**# from sklearn.pipeline import Pipeline**

**# numeric\_transformer = StandardScaler()**

**# categorical\_transformer = (OrdinalEncoder())**

**# preprocessor = ColumnTransformer(**

**# transformers=[**

**# ("num", numeric\_transformer, num\_col),**

**# ("cat", categorical\_transformer, cat\_col)**

**# ]**

**# )**

**# # Column\_Transf = ColumnTransformer([**

**# # ('trf',LabelEncoder(sparse=False,drop='first', handle\_unknown='ignore'),['batsman','bowler','team1', 'team2', 'venue'])**

**# # ]**

**# # ,remainder='passthrough')**

**# # pipe = Pipeline(steps=[**

**# # ('step1',trf),**

**# # ('step2',StandardScaler()),**

**# # ('step3',RandomForestRegressor())**

**# # ])**

**# In[43]:**

**# # Scale selected columns by index**

**# df\_sorted.iloc[:, 0:171] = df\_sorted.apply(LabelEncoder().fit\_transform(df\_sorted.iloc[:,0:171]))**

**# df\_sorted.iloc[:, 172:] = df\_sorted.apply(StandardScaler().fit\_transform(df\_sorted.iloc[:,172:]))**

**# In[38]:**

**#Label Encoding for object to numeric conversion**

**from sklearn.preprocessing import LabelEncoder**

**le = LabelEncoder()**

**for feat in cat\_col:**

**df\_sorted[feat] = le.fit\_transform(df\_sorted[feat].astype(str))**

**print (df\_sorted.info())**

**# In[39]:**

**df\_sorted**

**# In[40]:**

**df\_sorted.iloc[:, 172:] = StandardScaler().fit\_transform(df\_sorted.iloc[:,172:])**

**# In[41]:**

**df\_sorted**

**# In[48]:**

**# scaler = StandardScaler()**

**# lable = LabelEncoder()**

**# #df\_upsampled = df\_upsampled.apply(LabelEncoder().fit\_transform)**

**# # transform data**

**# #test = df\_sorted.apply(lambda x: scaler.fit\_transform(x) if (x.dtype != 'object') else**

**# lable.fit\_transform(x))**

**# #scaled = pd.DataFrame(test,columns=cols)**

**# scaled = pd.DataFrame(scaler.fit\_transform(df\_upsampled.select\_dtypes(exclude='object')), columns=cols)**

**# In[42]:**

**df\_sclaed =df\_sorted.drop('year', axis=1)**

**# In[43]:**

**df\_sclaed**

**# In[44]:**

**# fitting one of the model to get the feature which are having high impact**

**from sklearn.ensemble import RandomForestClassifier**

**from sklearn.model\_selection import train\_test\_split**

**df\_sclaed\_X = df\_sclaed.drop(['classification'], axis=1)**

**df\_sclaed\_y = df\_sclaed['classification']**

**# converting from DF to numpy before fitting**

**X=df\_sclaed\_X.values**

**y=df\_sclaed\_y.values**

**#X\_train, X\_test, y\_train, y\_test = train\_test\_split(df\_sclaed.drop(['classification'], axis=1), df\_sclaed['classification'],random\_state =42)**

**RFC = RandomForestClassifier(n\_jobs=-1, class\_weight='balanced', max\_depth=5)**

**RFC.fit(X, y)**

**# In[45]:**

**# dimensionality reduction using Boruta**

**from boruta import BorutaPy**

**# define Boruta feature selection method**

**feat\_selector = BorutaPy(RFC, n\_estimators='auto', verbose=2, random\_state=1)**

**# find all relevant features**

**feat\_selector.fit(X, y)**

**# Feature\_accept = X.columns[feat\_selector.support\_].to\_list()**

**# Feature\_tentative = X.columns[feat\_selector.support\_weak\_].to\_list()**

**# print('features in the accept area:', Feature\_accept)**

**# print('features in the tentative area:', Feature\_tentative)**

**# In[46]:**

**cols\_support =df\_sclaed\_X.columns[feat\_selector.support\_]**

**cols\_support**

**# In[47]:**

**# check selected features**

**feat\_selector.support\_**

**# check ranking of features**

**feat\_selector.ranking\_**

**# call transform() on X to filter it down to selected features**

**X\_filtered = feat\_selector.transform(X)**

**# In[48]:**

**# zip my names, ranks, and decisions in a single iterable**

**feature\_ranks = list(zip(df\_sclaed.columns,**

**feat\_selector.ranking\_,**

**feat\_selector.support\_))**

**# iterate through and print out the results**

**for feat in feature\_ranks:**

**print('Feature: {:<25} Rank: {}, Keep: {}'.format(feat[0], feat[1], feat[2]))**

**# In[56]:**

**# feat\_selector.support\_weak\_**

**# In[57]:**

**# test = X.values[feat\_selector.support\_].to\_list()**

**# In[58]:**

**# Df\_ = pd.DataFrame(X\_filtered)**

**# In[59]:**

**#!pip install Boruta**

**# In[49]:**

**X\_filtered = pd.DataFrame(X\_filtered,columns=cols\_support)**

**# In[50]:**

**X\_filtered**

**# In[51]:**

**# #lambda x: x.str.upper() if (x.dtype == 'object') else x**

**# scaler = StandardScaler()**

**# # transform data**

**# scaled = scaler.fit\_transform(data)**

**# import numpy as np**

**# from sklearn.preprocessing import LabelEncoder**

**# df\_upsampled = df\_upsampled.apply(LabelEncoder().fit\_transform)**

**# from sklearn.ensemble import RandomForestClassifier**

**from sklearn.model\_selection import train\_test\_split**

**X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_filtered, y,random\_state =42)**

**RFC = RandomForestClassifier()**

**RFC.fit(X\_train, y\_train)**

**# In[52]:**

**X\_filtered**

**# In[53]:**

**from sklearn.model\_selection import cross\_val\_score**

**clf = RandomForestClassifier()**

**# In[54]:**

**scores = cross\_val\_score(clf, X\_train, y\_train, cv=10)**

**# In[55]:**

**scores**

**# In[56]:**

**print("%0.22f accuracy with a standard deviation of %0.2f" % (scores.mean(), scores.std()))**

**# In[57]:**

**RFC.score(X\_test, y\_test)**

**# In[58]:**

**X\_filtered.shape**

**# In[59]:**

**def confmat(cm):**

**fig = plt.figure()**

**ax = fig.add\_subplot(111)**

**cax = ax.matshow(cm, cmap=plt.cm.Wistia)**

**plt.title('Model Confusion Matrix')**

**fig.colorbar(cax)**

**categories = ['Good','Not Good']**

**ax.set\_xticklabels([''] + categories)**

**ax.set\_yticklabels([''] + categories)**

**for i in range(2):**

**for j in range(2):**

**ax.text(i, j, cm[j, i], va='center', ha='center')**

**plt.xlabel('Predicted')**

**plt.ylabel('True')**

**plt.show()**

**# In[60]:**

**def featureimp(columns, importances):**

**importances = pd.DataFrame(data={**

**'Attribute': columns,**

**'Importance': importances**

**})**

**importances = importances.sort\_values(by='Importance', ascending=False)**

**plt.figure(figsize=(10, 10))**

**plt.bar(x=importances['Attribute'][1:20], height=importances['Importance'][1:20], color='salmon')**

**plt.title('Top 20 Impactful Feature', size=20)**

**plt.xticks(rotation='vertical')**

**plt.show()**

**# In[96]:**

**def partuning(x):**

**rscv\_fit = x.fit(X\_train, y\_train)**

**best\_parameters = rscv\_fit.best\_params\_**

**best\_estimator = rscv\_fit.best\_estimator\_**

**print(best\_parameters)**

**print(best\_estimator)**

**# In[98]:**

**from sklearn.metrics import classification\_report**

**from sklearn.model\_selection import RandomizedSearchCV**

**# In[100]:**

**rft = RandomForestClassifier()**

**grid\_rf = {**

**'n\_estimators': [20, 50, 100, 500, 1000],**

**'max\_depth': np.arange(1, 15, 1),**

**'min\_samples\_split': [2, 10, 9],**

**'min\_samples\_leaf': np.arange(1, 15, 2, dtype=int),**

**'bootstrap': [True, False],**

**'random\_state': [1, 2, 30, 42]**

**}**

**rscv = RandomizedSearchCV(estimator = rft,**

**param\_distributions = grid\_rf,**

**cv = 10,**

**n\_jobs=-1,**

**verbose=2,**

**n\_iter=48)**

**# In[101]:**

**partuning(rscv)**

**# In[102]:**

**RFC = RandomForestClassifier(max\_depth=6, min\_samples\_leaf=11, min\_samples\_split=10,random\_state=42)**

**RFC.fit(X\_train, y\_train)**

**predrf = RFC.predict(X\_test)**

**scorerf=accuracy\_score(y\_test,predrf)**

**print('Accuracy : %.5f'%scorerf)**

**# In[103]:**

**print(classification\_report(y\_test, predrf, target\_names = ['NOT GOOD','GOOD']))**

**# In[62]:**

**confm = confusion\_matrix(y\_test,predrf)**

**confmat(confm)**

**# In[63]:**

**rfimp = RFC.feature\_importances\_**

**xcolumn = X\_train.columns**

**featureimp(xcolumn, rfimp)**

**# In[75]:**

**# feat\_selector = BorutaPy(**

**# RFC,**

**# verbose=2,**

**# n\_estimators='auto',**

**# random\_state=1**

**# )**

**# In[76]:**

**# feat\_selector.fit(np.array(X\_train), np.array(y\_train))**

**# In[77]:**

**# print("\n------Support and Ranking for each feature------")**

**# for i in range(len(feat\_selector.support\_)):**

**# if feat\_selector.support\_[i]:**

**# print("Passes the test: ", X\_train.columns[i],**

**# " - Ranking: ", feat\_selector.ranking\_[i])**

**# else:**

**# print("Doesn't pass the test: ",**

**# X\_train.columns[i], " - Ranking: ", feat\_selector.ranking\_[i])**

**# ### Gradient Boost**

**# In[109]:**

**gbt = GradientBoostingClassifier()**

**grid\_gb = {**

**"n\_estimators":[5,50,250,500],**

**"max\_depth":[1,3,5,7,9],**

**"learning\_rate":[0.01,0.1,1,10,100]**

**}**

**rscv = RandomizedSearchCV(estimator = gbt,**

**param\_distributions = grid\_gb,**

**cv = 10,**

**n\_jobs=-1,**

**verbose=2,**

**n\_iter=48)**

**# In[110]:**

**partuning(rscv)**

**# In[111]:**

**gradientb = GradientBoostingClassifier(learning\_rate=1, max\_depth=7, n\_estimators=50)**

**gradientb.fit(X\_train, y\_train)**

**predgb = gradientb.predict(X\_test)**

**scoregb=accuracy\_score(y\_test,predgb)**

**print('Accuracy : %.5f'%scoregb)**

**# In[113]:**

**print(classification\_report(y\_test, predgb, target\_names = ['NOT GOOD','GOOD']))**

**# In[114]:**

**confm = confusion\_matrix(y\_test,predgb)**

**confmat(confm)**

**# In[115]:**

**gbimp = gradientb.feature\_importances\_**

**xcolumn = X\_train.columns**

**featureimp(xcolumn, gbimp)**

**# ### AdaBoostClassifier**

**# In[116]:**

**abt = AdaBoostClassifier()**

**grid\_ab = {**

**"n\_estimators":[5,50,250,500],**

**"learning\_rate":[0.01,0.1,1,10,100]**

**}**

**rscv = RandomizedSearchCV(estimator = abt,**

**param\_distributions = grid\_ab,**

**cv = 10,**

**n\_jobs=-1,**

**verbose=2,**

**n\_iter=48)**

**# In[117]:**

**partuning(rscv)**

**# In[118]:**

**adab = AdaBoostClassifier(n\_estimators = 50,learning\_rate=0.01)**

**adab.fit(X\_train, y\_train)**

**predab = adab.predict(X\_test)**

**scoreab=accuracy\_score(y\_test,predab)**

**print('Accuracy : %.2f'%scoreab)**

**# In[119]:**

**print(classification\_report(y\_test, predab, target\_names = ['NOT GOOD','GOOD']))**

**# In[68]:**

**confm = confusion\_matrix(y\_test,predab)**

**confmat(confm)**

**# In[69]:**

**abimp = adab.feature\_importances\_**

**xcolumn = X\_train.columns**

**featureimp(xcolumn, abimp)**

**# # Logistic Regression**

**# In[120]:**

**lrt = LogisticRegression()**

**grid\_lr = {**

**"solver":['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'],**

**"penalty":['none', 'l1', 'l2', 'elasticnet'],**

**"C" : [100, 10, 1.0, 0.1, 0.01]**

**}**

**rscv = RandomizedSearchCV(estimator = lrt,**

**param\_distributions = grid\_lr,**

**cv = 10,**

**n\_jobs=-1,**

**verbose=2,**

**n\_iter=48)**

**# In[121]:**

**partuning(rscv)**

**# In[122]:**

**logit = LogisticRegression(C=100, solver='newton-cg')**

**logit.fit(X\_train, y\_train)**

**predlg = logit.predict(X\_test)**

**scorelg = accuracy\_score(y\_test,predlg)**

**print('Accuracy : %.2f'%scorelg)**

**# In[123]:**

**print(classification\_report(y\_test, predlg, target\_names = ['NOT GOOD','GOOD']))**

**# In[124]:**

**confm = confusion\_matrix(y\_test,predlg)**

**confmat(confm)**

**# # GaussianNB**

**# In[73]:**

**gnb = GaussianNB()**

**gnb.fit(X\_train, y\_train)**

**prednb = gnb.predict(X\_test)**

**scorenb = accuracy\_score(y\_test,prednb)**

**print('Accuracy : %.2f'%scorenb)**

**# In[89]:**

**print(classification\_report(y\_test, prednb, target\_names = ['NOT GOOD','GOOD']))**

**# In[74]:**

**confm = confusion\_matrix(y\_test,prednb)**

**confmat(confm)**

**# # KN classifier**

**# In[130]:**

**knt = KNeighborsClassifier()**

**n\_neighbors = range(1, 21, 2)**

**weights = ['uniform', 'distance']**

**metric = ['euclidean', 'manhattan', 'minkowski']**

**grid\_kn = dict(n\_neighbors=n\_neighbors,weights=weights,metric=metric)**

**rscv = RandomizedSearchCV(estimator = knt,**

**param\_distributions = grid\_kn,**

**cv = 10,**

**n\_jobs=-1,**

**verbose=2,**

**n\_iter=48)**

**# In[131]:**

**partuning(rscv)**

**# In[132]:**

**knn = KNeighborsClassifier(metric='manhattan', n\_neighbors=1, weights='distance')**

**knn.fit(X\_train, y\_train)**

**predkn = knn.predict(X\_test)**

**scorekn = accuracy\_score(y\_test,predkn)**

**print('Accuracy : %.2f'%scorekn)**

**# In[133]:**

**print(classification\_report(y\_test, predkn, target\_names = ['NOT GOOD','GOOD']))**

**# In[134]:**

**confm = confusion\_matrix(y\_test,predkn)**

**confmat(confm)**

**# # SVM**

**# In[137]:**

**svcl = SVC()**

**param\_grid = {'C': [0.1, 1, 10, 100],**

**'gamma': [1, 0.1, 0.01, 0.001],**

**'kernel': ['rbf', 'poly', 'sigmoid']}**

**rscv = RandomizedSearchCV(estimator = svcl,**

**param\_distributions = param\_grid,**

**cv = 10,**

**n\_jobs=-1,**

**verbose=2,**

**n\_iter=48)**

**# In[138]:**

**partuning(rscv)**

**# In[139]:**

**svc = SVC(C=1, gamma=0.001, kernel='poly')**

**svc.fit(X\_train, y\_train)**

**predsvc = svc.predict(X\_test)**

**scoresvc = accuracy\_score(y\_test,predsvc)**

**print('Accuracy : %.2f'%scoresvc)**

**# In[140]:**

**print(classification\_report(y\_test, predsvc, target\_names = ['NOT GOOD','GOOD']))**

**# In[141]:**

**confm = confusion\_matrix(y\_test,predsvc)**

**confmat(confm)**

**# In[80]:**

**RFC.score(X\_test, y\_test)**

**# In[81]:**

**# prepare configuration for cross validation test harness**

**from sklearn import model\_selection**

**seed = 7**

**# prepare models**

**models = []**

**models.append(('LR', LogisticRegression()))**

**models.append(('LDA', RandomForestClassifier()))**

**models.append(('KNN', KNeighborsClassifier()))**

**models.append(('ADA', AdaBoostClassifier()))**

**models.append(('NB', GaussianNB()))**

**models.append(('SVM', SVC()))**

**models.append(('GR', GradientBoostingClassifier()))**

**# evaluate each model in turn**

**results = []**

**names = []**

**scoring = 'accuracy'**

**for name, model in models:**

**kfold = model\_selection.KFold(n\_splits=10, random\_state=seed, shuffle=True)**

**cv\_results = model\_selection.cross\_val\_score(model, X\_train, y\_train, cv=kfold, scoring=scoring)**

**results.append(cv\_results)**

**names.append(name)**

**msg = "%s: %f (%f)" % (name, cv\_results.mean(), cv\_results.std())**

**print(msg)**

**# boxplot algorithm comparison**

**fig = plt.figure()**

**fig.suptitle('Algorithm Comparison')**

**ax = fig.add\_subplot(111)**

**plt.boxplot(results)**

**ax.set\_xticklabels(names)**

**plt.show()**

**# In[ ]:**