



MLM-2

PROJECT 2 REPORT

CLASSIFICATION OF
CONSUMER DATA INTO
SEGMENTS FOR
INTERSTELLAR TOURISM

Submitted To: Prof. Amarnath Mitra

Submitted By:

Adhyatik (311063)

TABLE OF CONTENTS

1. COVER PAGE.....	1
2. TABLE OF CONTENTS.....	2
3. ACKNOWLEDGEMENT.....	3
4. OBJECTIVES.....	4
5. DESCRIPTION OF DATA.....	5
6. ANALYSIS OF DATA.....	14
7. OBSERVATIONS.....	51
8. MANAGERIAL INSIGHTS.....	87

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to Prof. Amarnath Mitra for his exceptional guidance and mentorship throughout the learning process of machine learning concepts. His profound knowledge, clear explanations, and dedication to teaching have been instrumental in my understanding and application of these concepts.

Prof. Amarnath Mitra's teachings have been invaluable in shaping my ability to analyze and interpret data effectively. His expertise in supervised machine learning techniques, particularly clustering algorithms like Decision Tree, Support Vector Machines, K Nearest Neighbor, Logistic Regression has been crucial in conducting this customer segmentation analysis for interstellar tourism.

I am grateful for the engaging and insightful lectures, where Prof. Mitra provided real-world examples and practical applications of machine learning concepts. His ability to break down complex topics into understandable components has been truly remarkable.

Furthermore, I would like to acknowledge Prof. Mitra's unwavering support and encouragement throughout this project. His invaluable feedback and suggestions have greatly contributed to the successful completion of this analysis and the generation of actionable insights.

I extend my heartfelt appreciation to Prof. Amarnath Mitra for imparting his knowledge and fostering a stimulating learning environment. His teachings have not only equipped me with the necessary skills but have also ignited a passion for exploring the vast potential of machine learning in solving complex business problems.

1.)OBJECTIVES

1.1) Classification of Consumer Data into Classes using Supervised Learning Classification Algorithms.

1.2.) Determination of an Appropriate Classification Model.

1.3.) Identification of Contributing Significant Variables and their Thresholds for Classification.

2.) DESCRIPTION OF DATA

2.1. Data Source, Size & Shape

2.1.1. Data Source (Website Link):-

<https://www.kaggle.com/datasets/anthonytherrien/interstellar-travel-customer-satisfaction-analysis>

2.1.2. Data Size:- 74.6 MB

2.1.3. Data Description & Dimensions:- This dataset, titled "Interstellar Travel Customer Satisfaction Analysis," provides a comprehensive view of customer experiences in interstellar space travel. It encompasses approximately **500,000** records, each representing an individual space travel experience. The primary objective of this dataset is to understand and predict the Customer Satisfaction Score, a key indicator of service quality and customer experience in the burgeoning field of interstellar tourism and travel.

Number Of Observations Taken For Analysis:- 1,00,192

Number Of Variables:- 18

2.2. Description of Variables

2.2.1. Index Variable(s): Row ID

2.2.2. Outcome Variable :- CLUSTER

2.2.3. Categorical Variables or Features (CV):- 09

1. **Gender:** Gender of the traveler.
2. **Occupation:** Occupation of the traveler, such as Colonist, Tourist, Businessperson, etc.
3. **Travel Class:** Class of travel, e.g., Business, Economy, Luxury.
4. **Destination:** Interstellar destination.
5. **Purpose of Travel:** The primary purpose of travel, e.g., Tourism, Research, Colonization.
6. **Transportation Type:** Type of transportation, e.g., Warp Drive, Solar Sailing, Ion Thruster.
7. **Special Requests:** Any special requests made by the traveler.
8. **Loyalty Program Member:** Indicates if the traveler is a member of a loyalty program.
9. **Month:** Month of travel.

2.2.3.1. Categorical Variables or Features - Nominal Type:- 09

1. **Gender**
2. **Occupation**
3. **Travel Class**
4. **Destination**
5. **Purpose of Travel**
6. **Transportation Type**
7. **Special Requests**
8. **Loyalty Program Member**
9. **Month**

2.2.3.2. Categorical Variables or Features - Ordinal Type: 0

There are no Categorical Variables of Ordinal Type in this Dataset.

2.2.4. Non-Categorical Variables or Features: 8

1. **Age:** Age of the traveler.
2. **Distance to Destination (Light-Years):** The distance to the destination measured in light-years.
3. **Duration of Stay (Earth Days):** Duration of stay at the destination in Earth days.
4. **Number of Companions:** The number of companions accompanying the traveler.
5. **Price (Galactic Credits):** Price of the trip in Galactic Credits.
6. **Booking Date:** Date when the trip was booked.
7. **Departure Date:** Date of departure.
8. **Customer Satisfaction Score:** Indicator of service quality and customer experience.

2.3. Descriptive Statistics

2.3.1. Descriptive Statistics: Outcome Variable (Categorical)

2.3.1.1. Count & Relative Frequency Statistics

Cluster	Count	Relative Frequency
cluster_2	34420	0.3435404024273395
cluster_0	32996	0.3293276908335995
cluster_1	32776	0.327131906739061

2.3.2. Descriptive Statistics: Categorical Variables or Features

2.3.2.1. Count & Relative Frequency Statistics

1. Gender Count Relative Frequency

Male	58403	0.5829108112424146
Female	41789	0.41708918875758544

2. Occupation Count Relative Frequency

Tourist	16768	0.16735867135100607
Businessperson	16756	0.1672389013094858
Scientist	16746	0.16713909294155221
Other	16666	0.1663406259980837
Colonist	16655	0.16623083679335676
Explorer	16601	0.1656918716065155

3. Travel Class Count Relative Frequency

Economy	54967	0.5486166560204407
Business	31828	0.3176700734589588
Luxury	13397	0.13371327052060045

4. Destination Count Relative Frequency

Alpha Centauri	9265	0.09247245289045034
Zeta II Reticuli	9187	0.09169394762056851
Proxima Centauri	9167	0.09149433088470137
Gliese 581	9146	0.09128473331204089
Tau Ceti	9146	0.09128473331204089
Trappist-1	9127	0.0910950974129671
Barnard's Star	9101	0.09083559565633982
Kepler-22b	9019	0.09001716703928457
Epsilon Eridani	9019	0.09001716703928457
Lalande 21185	8967	0.08949816352603002
Exotic Destination 9	986	0.00984110507824976
Exotic Destination 5	919	0.009172389013094858

Exotic Destination 4	914	0.009122484829128074
Exotic Destination 3	910	0.009082561481954647
Exotic Destination 2	901	0.008992733950814437
Exotic Destination 8	898	0.008962791440434366
Exotic Destination 1	893	0.008912887256467583
Exotic Destination 7	880	0.008783136378153944
Exotic Destination 6	879	0.008773155541360588
Exotic Destination 10	868	0.008663366336633664

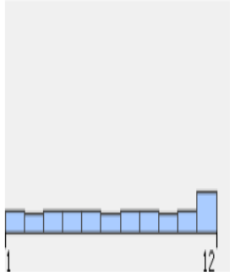
5. Purpose Of Travel	Count	Relative Frequency
Research	20352	0.2031299904183967
Other	20060	0.2002155860747365
Tourism	19935	0.1989679814755669
Colonization	19926	0.1988781539444267
Business	19919	0.1988082880868732

6. Transportation Type	Count	Relative Frequency
Solar Sailing	25154	0.25105796870009583
Warp Drive	25085	0.2503692909613542
Ion Thruster	24977	0.24929136058767168
Other	24976	0.2492813797508783

7. Special Requests	Count	Relative Frequency
Window Seat	20139	0.2010040721814117
Extra Space Suit	20101	0.20062480038326413
Special Meal	20017	0.19978641009262216
None	19991	0.1995269083359949
Other	19944	0.1990578090067071

8. Loyalty Status	Count	Relative Frequency
Yes	52117	0.520171271159374
No	48075	0.479828728840626

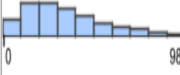
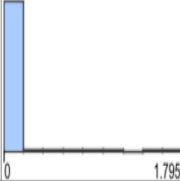
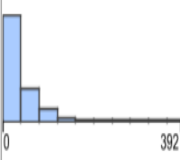
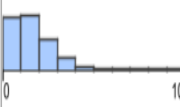
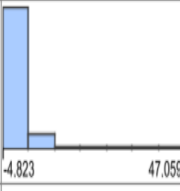
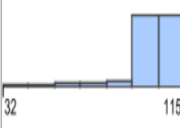
9. Month	Count	Relative Frequency
1	8744	0.08727243692111146
12	8596	0.08579527307569466
8	8548	0.08531619290961354
5	8544	0.08527626956244011
10	8525	0.08508663366336634
7	8496	0.084797189396359
3	8485	0.08468740019163207
4	8281	0.08265130948578729
9	8110	0.08094458639412329
11	8067	0.08051541041200894
6	8044	0.08028585116576174
2	7744	0.07729160012775471

Column	Min	Mean	Median	Max	Std. Dev.	Skewness	Kurtosis	No. Missing	No. +∞	No. -∞	Histogram
Month	1	6.5058	7	12	3.4582	-0.0041	-1.2112	8	0	0	

2.3.3. Descriptive Statistics: Non-Categorical Variables or Features

2.3.3.1. Measures of Central Tendency

2.3.3.2. Measures of Dispersion

Row ID	S Column	D Min	D Max	D Mean	D Std. de...	D Variance	D Skewness	D Kurtosis	D Overall ...	I No. mis...	I No. Na/Is	I No. +oss	I No. -oss	D Median	I Row co...	Histogram
Age	Age	0	98	31.184	19.677	387.188	0.723	-0.127	3,124,363	0	0	0	0	28	100192	
Distance to D...	Distance to ...	0	1,795.31	8.302	23.635	558.637	19.638	841.539	831,550.93	26	0	0	0	2.7	100192	
Duration of St...	Duration of ...	0	391.5	34.067	33.317	1,110.015	1.85	4.557	3,412,536	22	0	0	0	22	100192	
Number of Co...	Number of C...	0	10	1.108	1.095	1.199	1.108	1.577	111,043	1	0	0	0	1	100192	
Price (Galactic...	Price (Galact...	-4,822.526	47,059.009	1,025.638	1,366.348	1,866,908.213	4.843	57.855	102,699,18...	60	0	0	0	630.696	100192	
Customer Sati...	Customer S...	32.25	115	101.627	9.304	86.56	-2.372	7.715	10,178,498.88	37	0	0	0	102	100192	

2.3.3.3. Correlation Statistics

Correlation Matrix

Row ID	D Age	D Distance to Destina...	D Duration of Stay (...)	D Number of Comp...	D Price (Galactic Cre...	D Customer Satisfac...
Age	1.0	0.0012018622331481...	-3.268359978797662...	-0.07617321187515...	-0.13291710909193...	-0.19332349866477...
Distance t...	0.00120186...	1.0	0.06848002091921744	0.001791694565926...	0.5703616094405309	-0.05419203578003...
Duration o...	-3.2683599...	0.06848002091921744	1.0	-0.06400337965526...	0.049832633893204...	-0.00128230287324...
Number of...	-0.0761732...	0.0017916945659265...	-0.064003379655267...	1.0	0.008712973293811...	-0.02143670845884...
Price (Gala...	-0.1329171...	0.5703616094405309	0.04983263389320477	0.008712973293811...	1.0	0.090895137117804...
Customer ...	-0.1933234...	-0.05419203578003211	-0.001282302873247...	-0.02143670845884...	0.090895137117804...	1.0

Row ID	S First col...	S Second column name	D Correlation value	D p value	I Degrees...
Row ID 0	Age	Distance to Destination (Light-Years)	0.0012018622331481806	0.7036319749300362	100190
Row1	Age	Duration of Stay (Earth Days)	-3.268359978797662E-4	0.9176038447543221	100190
Row2	Age	Number of Companions	-0.07617321187515685	8.22863333694245...	100190
Row3	Age	Price (Galactic Credits)	-0.13291710909193286	0.0	100190
Row4	Age	Customer Satisfaction Score	-0.19332349866477952	0.0	100190
Row5	Distance to ...	Duration of Stay (Earth Days)	0.06848002091921744	0.0	100190
Row6	Distance to ...	Number of Companions	0.0017916945659265738	0.5706322419283532	100190
Row7	Distance to ...	Price (Galactic Credits)	0.5703616094405309	0.0	100190
Row8	Distance to ...	Customer Satisfaction Score	-0.05419203578003211	4.78262973117410...	100190
Row9	Duration of ...	Number of Companions	-0.06400337965526706	1.94679859254818...	100190
Row10	Duration of ...	Price (Galactic Credits)	0.04983263389320477	0.0	100190
Row11	Duration of ...	Customer Satisfaction Score	-0.0012823028732476...	0.6848278575279976	100190
Row12	Number of C...	Price (Galactic Credits)	0.008712973293811595	0.00581658376333...	100190
Row13	Number of C...	Customer Satisfaction Score	-0.021436708458846328	1.15219608378159...	100190
Row14	Price (Galact...	Customer Satisfaction Score	0.09089513711780459	0.0	100190

	Age	D...	D...	N...	P...	C...
Age	corr == -1 corr == +1 corr == n/a					
Distance to Desti...						
Duration of Stay ...						
Number of Comp...						
Price (Galactic Cr...						
Customer Satisfa...						

3.) ANALYSIS OF DATA

3.1. Data Pre-Processing

3.1.1. Missing Data Statistics and Treatment

3.1.1.1 Missing Data Statistics:

Field-wise Analysis:

1. **Distance To Destination** : 26 missing values
2. **Duration Of Stay** : 22 missing values
3. **Number Of Companions** : 1 missing values
4. **Price(Galactic Credits)** : 60 missing values
5. **Customer Satisfaction Score**: 37 missing values
6. **Month**:- 08 missing values

Distance To Destination, Duration Of Stay, Number Of Companions, Price, Customer Satisfaction Score All of these are NON-CATEGORICAL Variables and Month is a CATEGORICAL Variable.

Record-wise Analysis:

1. Total missing records: 154

3.1.1.2 Missing Data Treatment:

1. Since all the missing values are non-categorical, the treatment will focus mainly on non-categorical fields and also the treatment of the only categorical variable(Month).
2. Treatment of 'price' Variable: As 'price' is a non-categorical variable in numerical format, the Simple Imputer method is applied to all fields containing null values. The imputation strategy is 'mean', which replaces missing values with the mean value in each column. Similar Treatment is done for all the variables having missing values as all of them are Non-Categorical and have numerical values. Following the imputation process, no columns retain null values.
3. Treatment of 'Month' Variable: As 'Month' is a categorical variable in non-numerical format, the Simple Imputer method is applied to all fields containing null values. The imputation strategy is 'most frequent value', which replaces missing values with the mode value in each column.

3.1.1.3 Missing Data Exclusion:

1. Deletion of Records: If any Record(Rows) contains more than 50% of null values those Records are removed from the dataset.
2. Variable Deletion: If any Variable(Columns) contains more than 50% of null values persist, the corresponding variables are considered for removal.
3. Following the implementation of deletion procedures, the dataset achieves completeness.

3.1.2. Numerical Encoding of Categorical Variables or Features

Since all categorical variables in the dataset are nominal, we apply label encoding to transform them into numerical representations and the Encoding Schema is Alphanumeric Order.

3.1.3. Outlier Statistics and Treatment

3.1.3.1.1. Outlier Statistics: Non-Categorical Variables or Features

Row ID	S Outlier c...	I Member ...	I Outlier c...	D Lower bo...	D Upper bo...
Row0	Age	100192	922	-24.5	83.5
Row1	Distance to ...	100192	10948	-8.69	17.07
Row2	Duration of ...	100192	5158	-43	101
Row3	Number of ...	100192	235	-3	5
Row4	Price (Galac...	100192	7295	-1,268.819	2,832.369
Row5	Customer S...	100192	6289	89.5	117.5

3.1.3.1.2. Outlier Treatment: Non-Categorical Variables or Features

1. Identification of Outliers: Box plots are utilized to visually inspect the presence of outliers within the dataset.
2. Outlier Detection: Outliers are identified in the following columns: 'Distance To Destination', 'Duration Of Stay', 'Number Of Companions', 'Price', 'Customer Satisfaction Score'
3. Outlier Treatment: Min-Max Scaling normalization technique is applied to address outliers in the identified columns. Min-Max Normalization transforms x to x' by converting each value of features to a range between **0 and 1**, and this is also known as **(0–1) Normalization**. If the data has negative values the range would have been between **-1 and 1**.

The formula for Min-Max Normalization is:

The diagram shows the formula for Min-Max Normalization with arrows pointing from descriptive labels to the components of the formula. The formula is $x' = \frac{x - \min(x)}{\max(x) - \min(x)}$. The labels and their corresponding parts are: 'Normalized Value' points to x' ; 'Original Value' points to x ; 'Maximum Value of x' points to $\max(x)$; and 'Minimum Value of x' points to $\min(x)$.

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Formula for Min-Max Normalization

4. Min-Max Scaling normalizes the data and removes outliers. We utilize the interquartile range (IQR) from the 25th to the 75th percentile.

3.1.4. Data Bifurcation: Training & Testing Sets

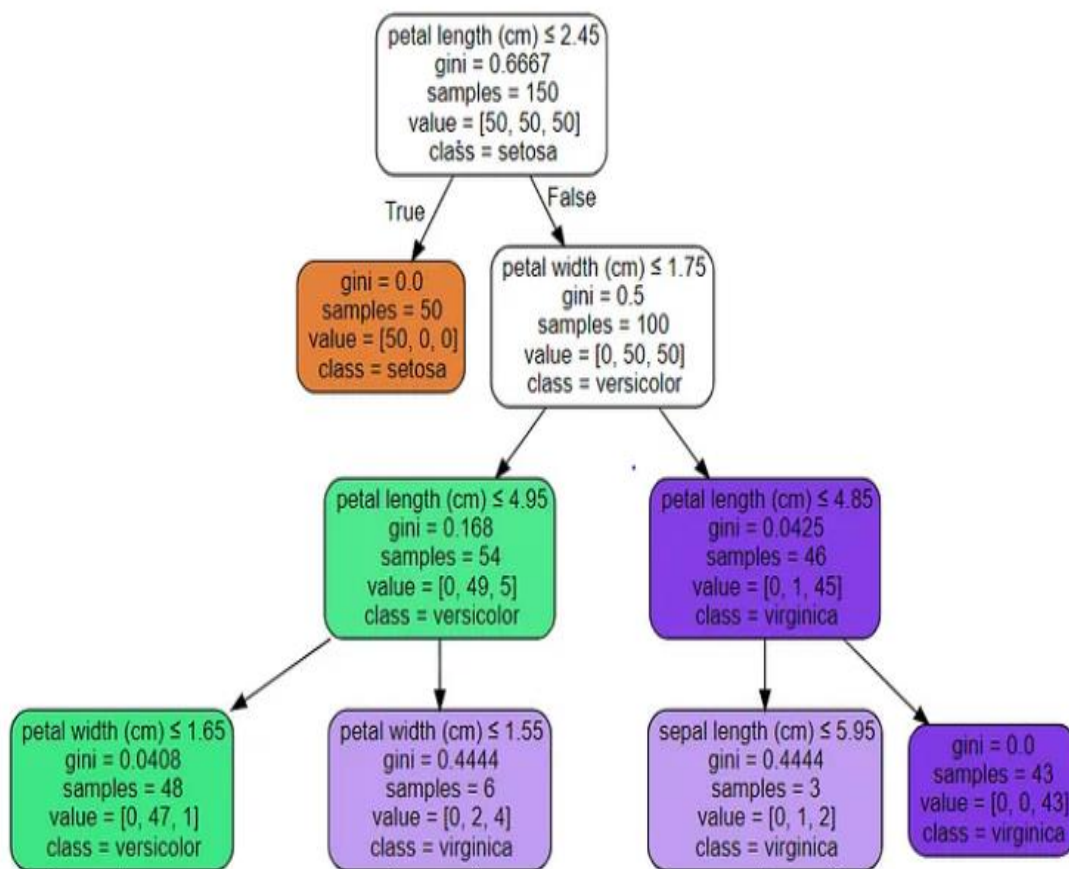
The Bifurcation Schema used is Stratified Sampling based on the Outcome Variable in the dataset i.e. Clusters and further 70% Data is allocated in the Training Set and 30% Data is allocated in the Testing Set.

3.2. Data Analysis

3.2.1. Supervised Machine Learning Classification Algorithm: Decision Tree (Base Model) |

Metrics Used - Gini Coefficient, Entropy

Decision Trees (DTs) are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. A tree can be seen as a piecewise constant approximation.



Understanding components of a Decision Tree

A decision tree is a branching flow diagram or tree chart. It comprises of the following components:

- A target variable such as diabetic or not and its initial distribution.
- A root node: this is the node that begins the splitting process by finding the variable that

best splits the target variable

- Node purity: Decision nodes are typically impure, or a mixture of both classes of the target variable (0,1 or green and red dots in the image). Pure nodes are those that have one class — hence the term pure. They either have green or red dots only in the image.
- Decision nodes: these are subsequent or intermediate nodes, where the target variable is again split further by other variables
- Leaf nodes or terminal nodes are pure nodes, hence are used for making a prediction of a numerical or class is made.

Some advantages of decision trees are:

- Simple to understand and to interpret. Trees can be visualized.
- Requires little data preparation. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed. Some tree and algorithm combinations support missing values.
- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data. However, the scikit-learn implementation does not support categorical variables for now. Other techniques are usually specialized in analyzing datasets that have only one type of variable.
- Able to handle multi-output problems.
- Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.
- Performs well even if its assumptions are somewhat violated by the true model from which the data were generated.

The disadvantages of decision trees include:

- Decision-tree learners can create over-complex trees that do not generalize the data well. This is called overfitting. Mechanisms such as pruning, setting the minimum number of

samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.

- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
- Predictions of decision trees are neither smooth nor continuous, but piecewise constant approximations as seen in the above figure. Therefore, they are not good at extrapolation.
- The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.
- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems.
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

In general, the run time cost to construct a balanced binary tree is $O(n_{\text{samples}} n_{\text{features}} \log(n_{\text{samples}}))$ and query time $O(\log(n_{\text{samples}}))$. Although the tree construction algorithm attempts to generate balanced trees, they will not always be balanced. Assuming that the subtrees remain approximately balanced, the cost at each node consists of searching through $O(n_{\text{features}})$ to find the feature that offers the largest reduction in the impurity criterion, e.g. log loss (which is equivalent to an information gain). This has a cost of $O(n_{\text{features}} n_{\text{samples}} \log(n_{\text{samples}}))$ at each node, leading to a total cost over the entire trees (by summing the cost at each node) of $O(n_{\text{features}} n_{\text{samples}}^2 \log(n_{\text{samples}}))$.

VARIOUS DECISION TREE ALGORITHMS

ID3 (Iterative Dichotomiser 3) was developed in 1986 by Ross Quinlan. The algorithm creates a multiway tree, finding for each node (i.e. in a greedy manner) the categorical feature that will yield the largest information gain for categorical targets. Trees are grown to their maximum size and then a pruning step is usually applied to improve the ability of the tree to generalize to unseen data. C4.5 is the successor to ID3 and removed the restriction that features must be categorical by dynamically defining a discrete attribute (based on numerical variables) that partitions the continuous attribute value into a discrete set of intervals. C4.5 converts the trained trees (i.e. the output of the ID3 algorithm) into sets of if-then rules. The accuracy of each rule is then evaluated to determine the order in which they should be applied. Pruning is done by removing a rule's precondition if the accuracy of the rule improves without it.

C5.0 is Quinlan's latest version release under a proprietary license. It uses less memory and builds smaller rulesets than C4.5 while being more accurate.

CART (Classification and Regression Trees) is very similar to C4.5, but it differs in that it supports numerical target variables (regression) and does not compute rule sets. CART constructs binary trees using the feature and threshold that yield the largest information gain at each node.

MATHEMATICAL FUNCTION :-

Given training vectors $x_i \in R^n$, $i=1, \dots, l$ and a label vector $y \in R^l$, a decision tree recursively partitions the feature space such that the samples with the same labels or similar target values are grouped together.

Let the data at node m be represented by Q_m with n_m samples. For each candidate split $\theta = (j, t_m)$ consisting of a feature j and threshold t_m , partition the data into $Q_m^{left}(\theta)$ and $Q_m^{right}(\theta)$ subsets

$$Q_m^{left}(\theta) = \{(x, y) | x_j \leq t_m\}$$
$$Q_m^{right}(\theta) = Q_m \setminus Q_m^{left}(\theta)$$

The quality of a candidate split of node m is then computed using an impurity function or loss function $H()$, the choice of which depends on the task being solved (classification or regression)

$$G(Q_m, \theta) = \frac{n_m^{left}}{n_m} H(Q_m^{left}(\theta)) + \frac{n_m^{right}}{n_m} H(Q_m^{right}(\theta))$$

Select the parameters that minimises the impurity

$$\theta^* = \operatorname{argmin}_{\theta} G(Q_m, \theta)$$

Recurse for subsets $Q_m^{left}(\theta^*)$ and $Q_m^{right}(\theta^*)$ until the maximum allowable depth is reached, $n_m < \min_{samples}$ or $n_m = 1$.

If a target is a classification outcome taking on values $0, 1, \dots, K-1$, for node m , let

$$p_{mk} = \frac{1}{n_m} \sum_{y \in Q_m} I(y = k)$$

be the proportion of class k observations in node m . If m is a terminal node, `predict_proba` for this region is set to p_{mk} . Common measures of impurity are the following.

Gini:

$$H(Q_m) = \sum_k p_{mk}(1 - p_{mk})$$

Log Loss or Entropy:

$$H(Q_m) = - \sum_k p_{mk} \log(p_{mk})$$

Variable selection criterion

Here is where the true complexity and sophistication of decision lies. Variables are selected on a complex statistical criterion which is applied at each decision node. Now, variable selection criterion in Decision Trees can be done via two approaches:

1. Entropy and Information Gain
2. Gini Index

Both criteria are broadly similar and seek to determine which variable would split the data to lead to the underlying child nodes being most homogenous or pure. Both are used in different Decision Tree algorithms. To add to the confusion, it is not clear which one is the preferred approach. So, one has to have an understanding of both.

ENTROPY AND INFORMATION GAIN

Entropy is a term that comes from physics and means a measure of disorder. More specifically, we can define it as:

Entropy is a scientific concept as well as a measurable physical property that is most commonly associated with a state of disorder, randomness, or uncertainty. The term and the concept are used in diverse fields, from classical thermodynamics, where it was first recognized, to the microscopic description of nature in statistical physics, and to the principles of information theory.

In information theory, the **entropy** of a random variable is the average level of “information”, “surprise”, or “uncertainty” inherent to the variable’s possible outcomes.

In the context of Decision Trees, entropy is a measure of disorder or impurity in a node. Thus, a node with more variable composition, such as 2 Pass and 2 Fail would be considered to have higher Entropy than a node which has only pass or only fail. The maximum level of entropy or disorder is given by 1 and minimum entropy is given by a value 0.

Leaf nodes which have all instances belonging to 1 class would have an entropy of 0. Whereas, the entropy for a node where the classes are divided equally would be 1.

Entropy is measured by the formula:

$$E = - \sum_{i=1}^n p_i \log_2(p_i)$$

Where the p_i is the probability of randomly selecting an example in class i .

Now essentially what a Decision Tree does to determine the root node is to calculate the entropy for each variable and its potential splits. For this we have to calculate a potential split from each

variable, calculate the average entropy across both or all the nodes and then the change in entropy vis a vis the parent node. This change in entropy is termed Information Gain and represents how much information a feature provides for the target variable.

$$\text{Information Gain} = \text{Entropy}_{\text{parent}} - \text{Entropy}_{\text{children}}$$

Entropy parent is the entropy of the parent node and Entropy children represents the average entropy of the child nodes that follow this variable.

GINI INDEX

The other way of splitting a decision tree is via the Gini Index. The Entropy and Information Gain method focuses on purity and impurity in a node. The Gini Index or Impurity measures the probability for a random instance being misclassified when chosen randomly. The lower the Gini Index, the better the lower the likelihood of misclassification.

The formula for Gini Index

$$Gini = 1 - \sum_{i=1}^j P(i)^2$$

Where j represents the no. of classes in the target variable — Pass and Fail in our example

P(i) represents the ratio of Pass/Total no. of observations in node.

Gini Index vs Information Gain

Depending on which impurity measurement is used, tree classification results can vary. This can make small (or sometimes large) impact on your model. There seems to be no one preferred approach by different Decision Tree algorithms. For example, CART uses Gini; ID3 and C4.5 use Entropy.

The Gini index has a maximum impurity is 0.5 and maximum purity is 0, whereas Entropy has a maximum impurity of 1 and maximum purity is 0.

3.2.2. Supervised Machine Learning Classification Algorithms: {Logistic Regression | Support Vector Machine | K Nearest Neighbour} (Comparison Models)

LOGISTIC REGRESSION

Logistic regression is a supervised machine learning algorithm that accomplishes binary classification tasks by predicting the probability of an outcome, event, or observation. The model delivers a binary or dichotomous outcome limited to two possible outcomes: yes/no, 0/1, or true/false.

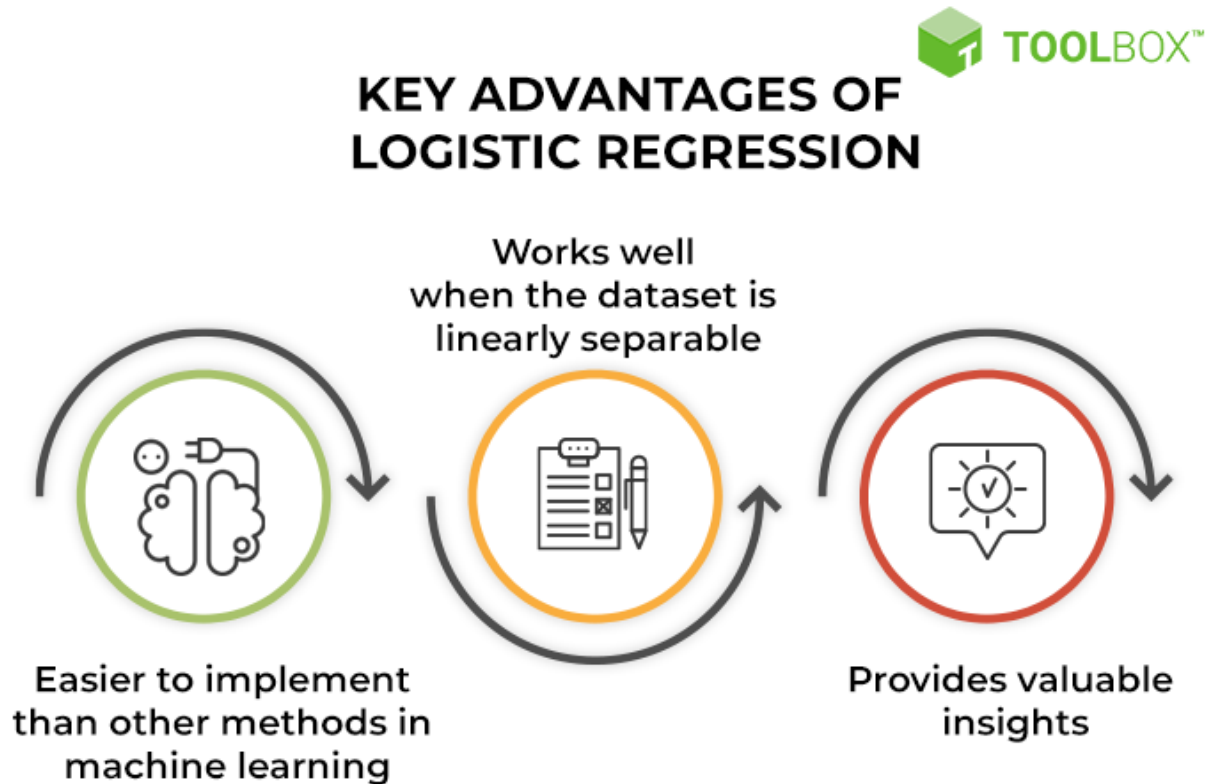
Logical regression analyzes the relationship between one or more independent variables and classifies data into discrete classes. It is extensively used in predictive modeling, where the model estimates the mathematical probability of whether an instance belongs to a specific category or not. For example, 0 – represents a negative class; 1 – represents a positive class. Logistic regression is commonly used in binary classification problems where the outcome variable reveals either of the two categories (0 and 1).

Some examples of such classifications and instances where the binary response is expected or implied are:

- 1. Determine the probability of heart attacks:** With the help of a logistic model, medical practitioners can determine the relationship between variables such as the weight, exercise, etc., of an individual and use it to predict whether the person will suffer from a heart attack or any other medical complication.
- 2. Possibility of enrolling into a university:** Application aggregators can determine the probability of a student getting accepted to a particular university or a degree course in a college by studying the relationship between the estimator variables, such as GRE, GMAT, or TOEFL scores.
- 3. Identifying spam emails:** Email inboxes are filtered to determine if the email communication is promotional/spam by understanding the predictor variables and applying a logistic regression algorithm to check its authenticity.

Key advantages of logistic regression

The logistic regression analysis has several advantages in the field of machine learning.



1. Easier to implement machine learning methods: A machine learning model can be effectively set up with the help of training and testing. The training identifies patterns in the input data (image) and associates them with some form of output (label). Training a logistic model with a regression algorithm does not demand higher computational power. As such, logistic regression is easier to implement, interpret, and train than other ML methods.

2. Suitable for linearly separable datasets: A linearly separable dataset refers to a graph where a straight line separates the two data classes. In logistic regression, the y variable takes only two values. Hence, one can effectively classify data into two separate classes if linearly separable data is used.

3. Provides valuable insights: Logistic regression measures how relevant or appropriate an independent/predictor variable is (coefficient size) and also reveals the direction of their relationship or association (positive or negative).

Logistic Regression Equation and Assumptions

Logistic regression uses a logistic function called a sigmoid function to map predictions and their probabilities. The sigmoid function refers to an S-shaped curve that converts any real value to a range between 0 and 1.

Moreover, if the output of the sigmoid function (estimated probability) is greater than a predefined threshold on the graph, the model predicts that the instance belongs to that class. If the estimated probability is less than the predefined threshold, the model predicts that the instance does not belong to the class.

For example, if the output of the sigmoid function is above 0.5, the output is considered as 1. On the other hand, if the output is less than 0.5, the output is classified as 0. Also, if the graph goes further to the negative end, the predicted value of y will be 0 and vice versa. In other words, if the output of the sigmoid function is 0.65, it implies that there are 65% chances of the event occurring; a coin toss, for example.

The sigmoid function is referred to as an activation function for logistic regression and is defined as:

$$f(x) = \frac{1}{1 + e^{-x}}$$

Equation of Logistic Regression

where,

- e = base of natural logarithms
- value = numerical value one wishes to transform

The following equation represents logistic regression:



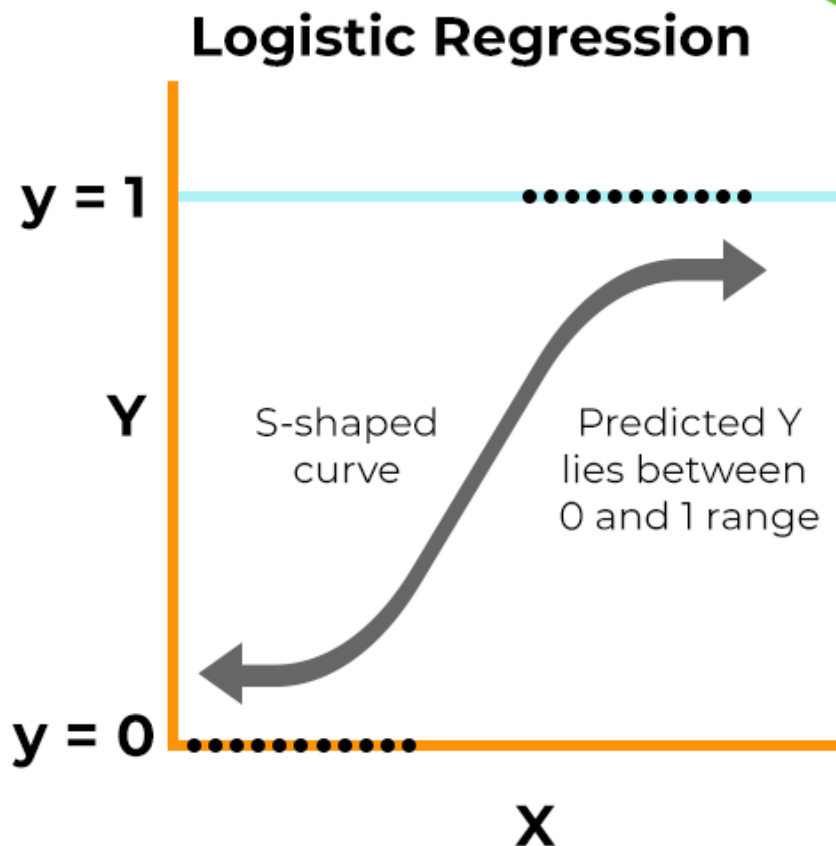
$$y = \frac{e^{(b_0 + b_1X)}}{1 + e^{(b_0 + b_1X)}}$$

Logistic Regression – Sigmoid Function

here,

- x = input value
- y = predicted output
- b0 = bias or intercept term
- b1 = coefficient for input (x)

This equation is similar to linear regression, where the input values are combined linearly to predict an output value using weights or coefficient values. However, unlike linear regression, the output value modeled here is a binary value (0 or 1) rather than a numeric value.



Key Assumptions for Implementing Logistic Regression

Key properties of the logistic regression equation

Typical properties of the logistic regression equation include:

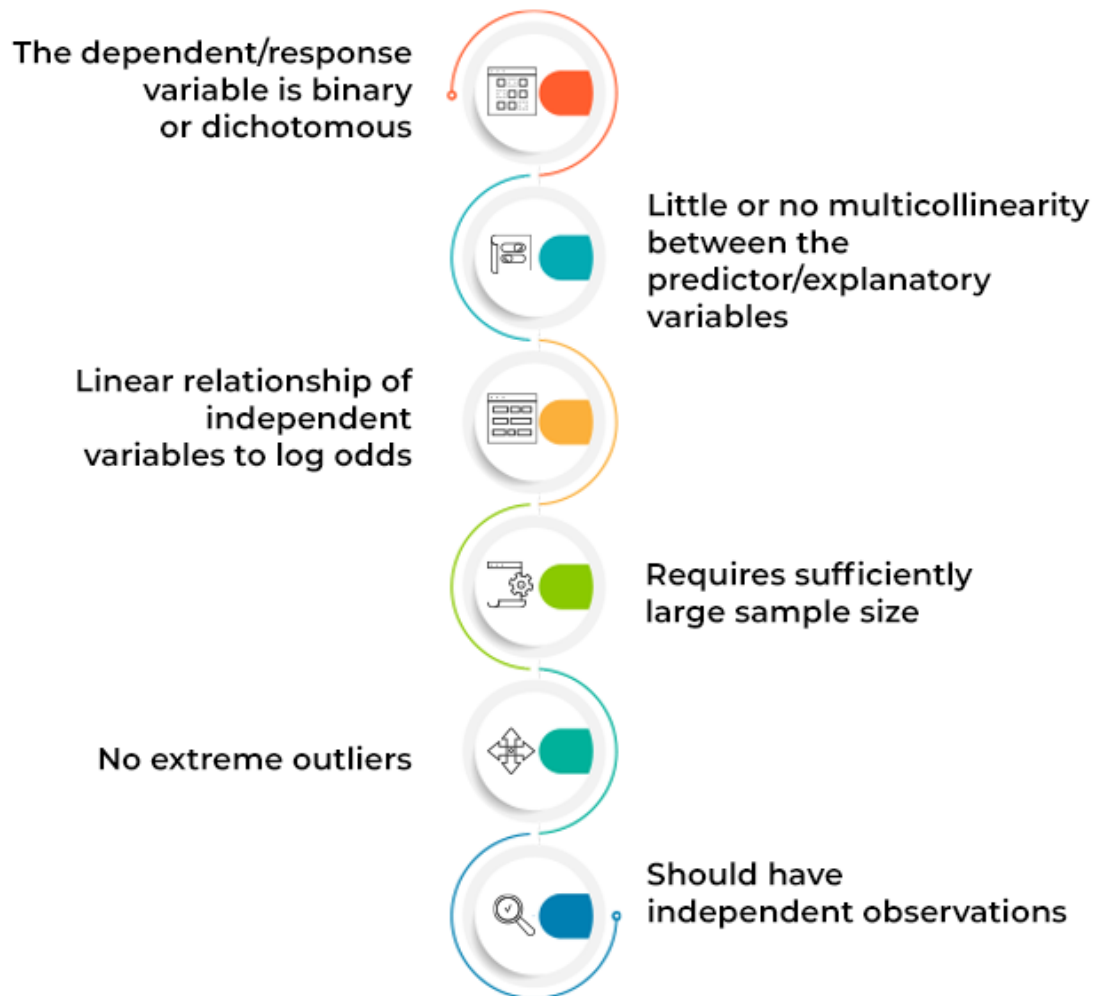
- Logistic regression's dependent variable obeys 'Bernoulli distribution'
- Estimation/prediction is based on 'maximum likelihood.'
- Logistic regression does not evaluate the coefficient of determination (or R squared) as observed in linear regression'. Instead, the model's fitness is assessed through a concordance.

For example, KS or Kolmogorov-Smirnov statistics look at the difference between cumulative events and cumulative non-events to determine the efficacy of models through credit scoring.

While implementing logistic regression, one needs to keep in mind the following key assumptions:



KEY ASSUMPTIONS FOR IMPLEMENTING LOGISTIC REGRESSION



Logistic Regression Best Practices

1. The dependent/response variable is binary or dichotomous

The first assumption of logistic regression is that response variables can only take on two possible outcomes – pass/fail, male/female, and malignant/benign.

This assumption can be checked by simply counting the unique outcomes of the dependent variable. If more than two possible outcomes surface, then one can consider that this assumption is

violated.

2. Little or no multicollinearity between the predictor/explanatory variables

This assumption implies that the predictor variables (or the independent variables) should be independent of each other. Multicollinearity relates to two or more highly correlated independent variables. Such variables do not provide unique information in the regression model and lead to wrongful interpretation.

The assumption can be verified with the variance inflation factor (VIF), which determines the correlation strength between the independent variables in a regression model.

3. Linear relationship of independent variables to log odds

Log odds refer to the ways of expressing probabilities. Log odds are different from probabilities. Odds refer to the ratio of success to failure, while probability refers to the ratio of success to everything that can occur.

For example, consider that you play twelve tennis games with your friend. Here, the odds of you winning are 5 to 7 (or $5/7$), while the probability of you winning is 5 to 12 (as the total games played = 12).

4. Prefers large sample size

Logistic regression analysis yields reliable, robust, and valid results when a larger sample size of the dataset is considered.

This assumption can be validated by taking into account a minimum of 10 cases considering the least frequent outcome for each estimator variable. Let's consider a case where you have three predictor variables, and the probability of the least frequent outcome is 0.30. Here, the sample size would be $(10 \times 3) / 0.30 = 100$.

5. Problem with extreme outliers

Another critical assumption of logistic regression is the requirement of no extreme outliers in the dataset.

This assumption can be verified by calculating Cook's distance (D_i) for each observation to identify influential data points that may negatively affect the regression model. In situations when outliers exist, one can implement the following solutions:

- Eliminate or remove the outliers
- Consider a value of mean or median instead of outliers, or
- Keep the outliers in the model but maintain a record of them while reporting the regression results.

6. Consider independent observations

This assumption states that the dataset observations should be independent of each other. The observations should not be related to each other or emerge from repeated measurements of the same individual type.

The assumption can be verified by plotting residuals against time, which signifies the order of observations. The plot helps in determining the presence or absence of a random pattern. If a random pattern is present or detected, this assumption may be considered violated.

Types of Logistic Regression with Examples

Logistic regression is classified into binary, multinomial, and ordinal. Each type differs from the other in execution and theory. Let's understand each type in detail.

1. Binary logistic regression

Binary logistic regression predicts the relationship between the independent and binary dependent variables. Some examples of the output of this regression type may be, success/failure, 0/1, or true/false.

Examples:

1. Deciding on whether or not to offer a loan to a bank customer: Outcome = yes or no.
2. Evaluating the risk of cancer: Outcome = high or low.
3. Predicting a team's win in a football match: Outcome = yes or no.

2. Multinomial logistic regression

A categorical dependent variable has two or more discrete outcomes in a multinomial regression type. This implies that this regression type has more than two possible outcomes.

Examples:

1. Let's say you want to predict the most popular transportation type for 2040. Here, transport

type equates to the dependent variable, and the possible outcomes can be electric cars, electric trains, electric buses, and electric bikes.

2. Predicting whether a student will join a college, vocational/trade school, or corporate industry.
3. Estimating the type of food consumed by pets, the outcome may be wet food, dry food, or junk food.

3. Ordinal logistic regression

Ordinal logistic regression applies when the dependent variable is in an ordered state (i.e., ordinal).

The dependent variable (y) specifies an order with two or more categories or levels.

Examples: Dependent variables represent,

1. Formal shirt size: Outcomes = XS/S/M/L/XL
2. Survey answers: Outcomes = Agree/Disagree/Unsure
3. Scores on a math test: Outcomes = Poor/Average/Good

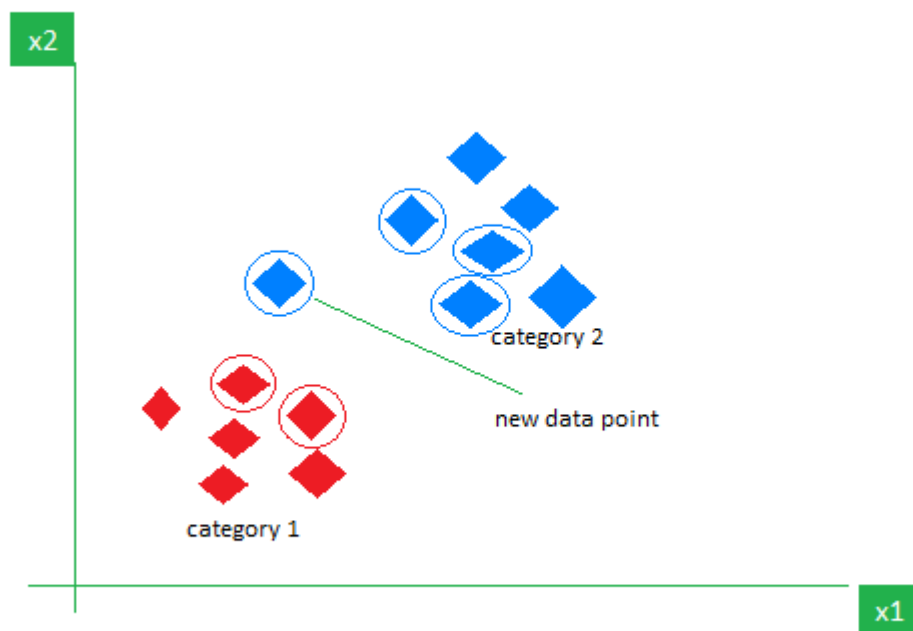
K-Nearest Neighbor(KNN)

The **K-Nearest Neighbors (KNN) algorithm** is a supervised machine learning method employed to tackle classification and regression problems. Evelyn Fix and Joseph Hodges developed this algorithm in 1951, which was subsequently expanded by Thomas Cover. The article explores the fundamentals, workings, and implementation of the KNN algorithm.

KNN is one of the most basic yet essential classification algorithms in machine learning. It belongs to the supervised learning domain and finds intense application in pattern recognition, data mining, and intrusion detection.

It is widely disposable in real-life scenarios since it is non-parametric, meaning it does not make any underlying assumptions about the distribution of data (as opposed to other algorithms such as GMM, which assume a Gaussian distribution of the given data). We are given some prior data (also called training data), which classifies coordinates into groups identified by an attribute.

As an example, consider the following table of data points containing two features:



KNN Algorithm working visualization

Now, given another set of data points (also called testing data), allocate these points to a group by analyzing the training set. Note that the unclassified points are marked as 'White'.

Intuition Behind KNN Algorithm

If we plot these points on a graph, we may be able to locate some clusters or groups. Now, given an unclassified point, we can assign it to a group by observing what group its nearest neighbors belong to. This means a point close to a cluster of points classified as 'Red' has a higher probability of getting classified as 'Red'.

Intuitively, we can see that the first point (2.5, 7) should be classified as 'Green', and the second point (5.5, 4.5) should be classified as 'Red'.

Why do we need a KNN algorithm?

(K-NN) algorithm is a versatile and widely used machine learning algorithm that is primarily used for its simplicity and ease of implementation. It does not require any assumptions about the underlying data distribution. It can also handle both numerical and categorical data, making it a flexible choice for various types of datasets in classification and regression tasks. It is a non-parametric method that makes predictions based on the similarity of data points in a given dataset. K-NN is less sensitive to outliers compared to other algorithms.

The K-NN algorithm works by finding the K nearest neighbors to a given data point based on a distance metric, such as Euclidean distance. The class or value of the data point is then determined by the majority vote or average of the K neighbors. This approach allows the algorithm to adapt to different patterns and make predictions based on the local structure of the data.

Distance Metrics Used in KNN Algorithm

As we know that the KNN algorithm helps us identify the nearest points or the groups for a query point. But to determine the closest groups or the nearest points for a query point we need some metric. For this purpose, we use below distance metrics:

Euclidean Distance

This is nothing but the cartesian distance between the two points which are in the plane/hyperplane. Euclidean distance can also be visualized as the length of the straight line that joins the two points which are into consideration. This metric helps us calculate the net displacement done between the two states of an object.

$$\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^d (x_j - X_{i_j})^2}$$

Manhattan Distance

Manhattan Distance metric is generally used when we are interested in the total distance traveled by the object instead of the displacement. This metric is calculated by summing the absolute difference between the coordinates of the points in n-dimensions.

$$d(x, y) = \sum_{i=1}^n |x_i - y_i|$$

Minkowski Distance

We can say that the Euclidean, as well as the Manhattan distance, are special cases of the Minkowski distance.

$$d(x, y) = \left(\sum_{i=1}^n (x_i - y_i)^p \right)^{\frac{1}{p}}$$

From the formula above we can say that when $p = 2$ then it is the same as the formula for the Euclidean distance and when $p = 1$ then we obtain the formula for the Manhattan distance.

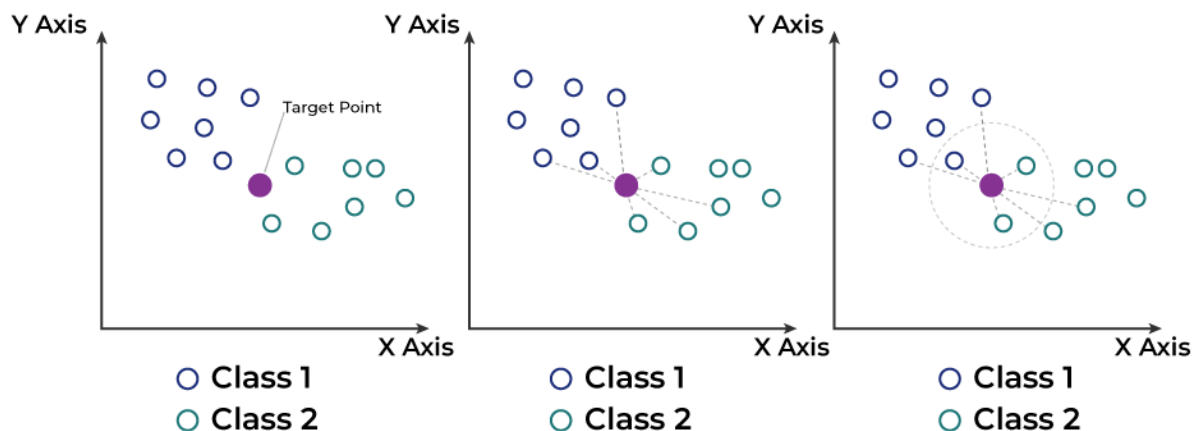
The above-discussed metrics are most common while dealing with a Machine Learning problem but there are other distance metrics as well like Hamming Distance which come in handy while dealing with problems that require overlapping comparisons between two vectors whose contents can be Boolean as well as string values.

How to choose the value of k for KNN Algorithm?

The value of k is very crucial in the KNN algorithm to define the number of neighbors in the algorithm. The value of k in the k-nearest neighbors (k-NN) algorithm should be chosen based on the input data. If the input data has more outliers or noise, a higher value of k would be better. It is recommended to choose an odd value for k to avoid ties in classification. Cross-validation methods can help in selecting the best k value for the given dataset.

Workings of KNN algorithm

The K-Nearest Neighbors (KNN) algorithm operates on the principle of similarity, where it predicts the label or value of a new data point by considering the labels or values of its K nearest neighbors in the training dataset.



Step-by-Step explanation of how KNN works is discussed below:

Step 1: Selecting the optimal value of K

- K represents the number of nearest neighbors that needs to be considered while making prediction.

Step 2: Calculating distance

- To measure the similarity between target and training data points, Euclidean distance is used. Distance is calculated between each of the data points in the dataset and target point.

Step 3: Finding Nearest Neighbors

- The k data points with the smallest distances to the target point are the nearest neighbors.

Step 4: Voting for Classification or Taking Average for Regression

- In the classification problem, the class labels are determined by performing majority voting. The class with the most occurrences among the neighbors becomes the predicted class for the target data point.
- In the regression problem, the class label is calculated by taking average of the target values of K nearest neighbors. The calculated average value becomes the predicted output for the target data point.

Let X be the training dataset with n data points, where each data point is represented by a d -dimensional feature vector and Y be the corresponding labels or values for each data point in X .

Given a new data point x , the algorithm calculates the distance between x and each data point in X using a distance metric, such as Euclidean distance. The algorithm selects the K data points from X that have the shortest distances to x . For classification tasks, the algorithm assigns the label y that is most frequent among the K nearest neighbors to x . For regression tasks, the algorithm calculates the average or weighted average of the values y of the K nearest neighbors and assigns it as the predicted value for x .

Advantages of the KNN Algorithm

- **Easy to implement** as the complexity of the algorithm is not that high.
- **Adapts Easily** – As per the working of the KNN algorithm it stores all the data in memory storage and hence whenever a new example or data point is added then the algorithm adjusts itself as per that new example and has its contribution to the future predictions as well.
- **Few Hyperparameters** – The only parameters which are required in the training of a KNN algorithm are the value of k and the choice of the distance metric which we would like to choose from our evaluation metric.

Disadvantages of the KNN Algorithm

- **Does not scale** – As we have heard about this that the KNN algorithm is also considered a Lazy Algorithm. The main significance of this term is that this takes lots of computing power as well as data storage. This makes this algorithm both time-consuming and resource exhausting.
- **Curse of Dimensionality** – There is a term known as the peaking phenomenon according to this the KNN algorithm is affected by the curse of dimensionality which implies the algorithm faces a hard time classifying the data points properly when the dimensionality is too high.
- **Prone to Overfitting** – As the algorithm is affected due to the curse of dimensionality it is prone to the problem of overfitting as well. Hence generally feature selection as well as dimensionality reduction techniques are applied to deal with this problem.

SUPPORT VECTOR MACHINE(SVM)

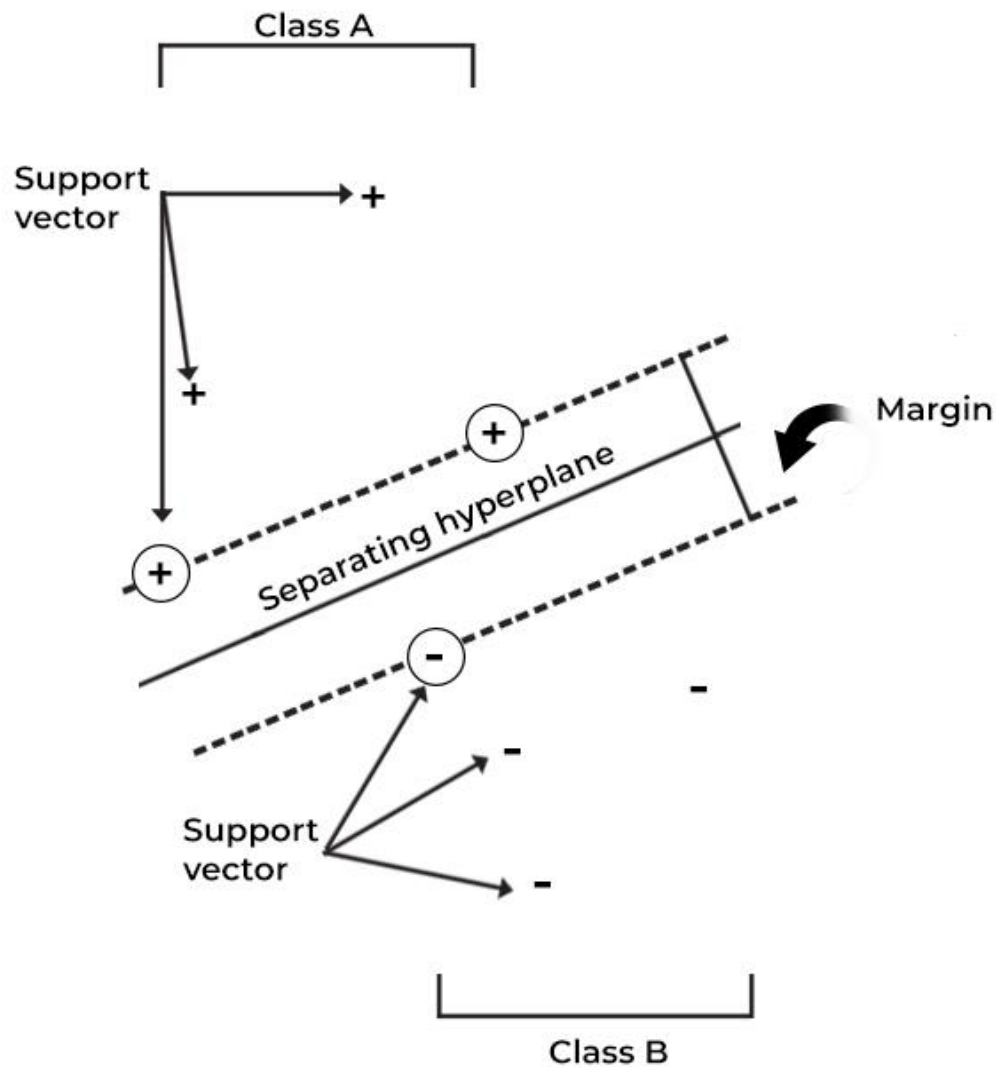
A support vector machine (SVM) is a machine learning algorithm that uses supervised learning models to solve complex classification, regression, and outlier detection problems by performing optimal data transformations that determine boundaries between data points based on predefined classes, labels, or outputs. SVMs are widely adopted across disciplines such as healthcare, natural language processing, signal processing applications, and speech & image recognition fields.

Technically, the primary objective of the SVM algorithm is to identify a hyperplane that distinguishably segregates the data points of different classes. The hyperplane is localized in such a manner that the largest margin separates the classes under consideration.

The support vector representation is shown in the figure below:



SVMS OPTIMIZE MARGIN BETWEEN SUPPORT VECTORS OR CLASSES



SVMs Optimize Margin Between Support Vectors or Classes

As seen in the above figure, the margin refers to the maximum width of the slice that runs parallel to the hyperplane without any internal support vectors. Such hyperplanes are easier to define for linearly separable problems; however, for real-life problems or scenarios, the SVM algorithm tries to maximize the margin between the support vectors, thereby giving rise to incorrect classifications for smaller sections of data points.

SVMs are potentially designed for binary classification problems. However, with the rise in computationally intensive multiclass problems, several binary classifiers are constructed and combined to formulate SVMs that can implement such multiclass classifications through binary means.

In the mathematical context, an SVM refers to a set of ML algorithms that use kernel methods to transform data features by employing kernel functions. Kernel functions rely on the process of mapping complex datasets to higher dimensions in a manner that makes data point separation easier. The function simplifies the data boundaries for non-linear problems by adding higher dimensions to map complex data points.

While introducing additional dimensions, the data is not entirely transformed as it can act as a computationally taxing process. This technique is usually referred to as the kernel trick, wherein data transformation into higher dimensions is achieved efficiently and inexpensively.

The idea behind the SVM algorithm was first captured in 1963 by Vladimir N. Vapnik and Alexey Ya. Chervonenkis. Since then, SVMs have gained enough popularity as they have continued to have wide-scale implications across several areas, including the protein sorting process, text categorization, facial recognition, autonomous cars, robotic systems, and so on.

How Does a Support Vector Machine Work?

The working of a support vector machine can be better understood through an example. Let's assume we have red and black labels with the features denoted by x and y . We intend to have a classifier for these tags that classifies data into either the red or black category.

Let's plot the labeled data on an x - y plane, as below:

A typical SVM separates these data points into red and black tags using the hyperplane, which is a two-dimensional line in this case. The hyperplane denotes the decision boundary line, wherein data

points fall under the red or black category.

A hyperplane is defined as a line that tends to widen the margins between the two closest tags or labels (red and black). The distance of the hyperplane to the most immediate label is the largest, making the data classification easier.

The above scenario is applicable for linearly separable data. However, for non-linear data, a simple straight line cannot separate the distinct data points.

Here's an example of non-linear complex dataset data:

The above dataset reveals that a single hyperplane is not sufficient to separate the involved labels or tags. However, here, the vectors are visibly distinct, making segregating them easier.

For data classification, you need to add another dimension to the feature space. For linear data discussed until this point, two dimensions of x and y were sufficient. In this case, we add a z -dimension to better classify the data points. Moreover, for convenience, let's use the equation for a circle, $z = x^2 + y^2$.

With the third dimension, the slice of feature space along the z -direction looks like this:

Now, with three dimensions, the hyperplane, in this case, runs parallel to the x -direction at a particular value of z ; let's consider it as $z=1$.

The remaining data points are further mapped back to two dimensions.

The above figure reveals the boundary for data points along features x , y , and z along a circle of the circumference with radii of 1 unit that segregates two labels of tags via the SVM.

Let's consider another method of visualizing data points in three dimensions for separating two tags (two different colored tennis balls in this case). Consider the balls lying on a 2D plane surface. Now, if we lift the surface upward, all the tennis balls are distributed in the air. The two differently colored balls may separate in the air at one point in this process. While this occurs, you can use or place the surface between two segregated sets of balls.

In this entire process, the act of 'lifting' the 2D surface refers to the event of mapping data into higher dimensions, which is technically referred to as 'kerneling', as mentioned earlier. In this way, complex data points can be separated with the help of more dimensions. The concept highlighted here is that the data points continue to get mapped into higher dimensions until a hyperplane is identified that shows a clear separation between the data points.

Types of Support Vector Machines

Support vector machines are broadly classified into two types: simple or linear SVM and kernel or non-linear SVM.

1. Simple or linear SVM

A linear SVM refers to the SVM type used for classifying linearly separable data. This implies that when a dataset can be segregated into categories or classes with the help of a single straight line, it is termed a linear SVM, and the data is referred to as linearly distinct or separable. Moreover, the classifier that classifies such data is termed a linear SVM classifier.

A simple SVM is typically used to address classification and regression analysis problems.

2. Kernel or non-linear SVM

Non-linear data that cannot be segregated into distinct categories with the help of a straight line is classified using a kernel or non-linear SVM. Here, the classifier is referred to as a non-linear classifier. The classification can be performed with a non-linear data type by adding features into higher dimensions rather than relying on 2D space. Here, the newly added features fit a hyperplane that helps easily separate classes or categories.

Kernel SVMs are typically used to handle optimization problems that have multiple variables.

3.2.3. Classification Model Performance Evaluation: Confusion Matrix {Accuracy, Recall, Precision, F1-Score} (Base Model: Decision Tree)

A confusion matrix is a performance evaluation tool in machine learning, representing the accuracy of a classification model. It displays the number of true positives, true negatives, false positives, and false negatives. This matrix aids in analyzing model performance, identifying misclassifications, and improving predictive accuracy.

A Confusion matrix is an $N \times N$ matrix used for evaluating the performance of a classification model, where N is the total number of target classes. The matrix compares the actual target values with those predicted by the machine learning model. This gives us a holistic view of how well our

classification model is performing and what kinds of errors it is making.

For a binary classification problem, we would have a **2 x 2 matrix**, as shown below, with 4 values:

		ACTUAL VALUES	
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	TP	FP
	NEGATIVE	FN	TN

Let's decipher the matrix:

- The target variable has two values: **Positive** or **Negative**
- The **columns** represent the **actual values** of the target variable
- The **rows** represent the **predicted values** of the target variable

But wait – what's TP, FP, FN, and TN here? That's the crucial part of a confusion matrix. Let's understand each term below.

Important Terms in a Confusion Matrix:-

True Positive (TP)

- The predicted value matches the actual value, or the predicted class matches the actual class.
- The actual value was positive, and the model predicted a positive value.

True Negative (TN)

- The predicted value matches the actual value, or the predicted class matches the actual class.
- The actual value was negative, and the model predicted a negative value.

False Positive (FP) – Type I Error

- The predicted value was falsely predicted.
- The actual value was negative, but the model predicted a positive value.
- Also known as the type I error.

False Negative (FN) – Type II Error

- The predicted value was falsely predicted.
- The actual value was positive, but the model predicted a negative value.
- Also known as the type II error.

Let me give you an example to better understand this. Suppose we had a classification dataset with 1000 data points. We fit a classifier (say logistic regression or decision tree) on it and get the below confusion matrix:

		ACTUAL VALUES	
		POSITIVE	NEGATIVE
PREDICTED VALUES	POSITIVE	560	60
	NEGATIVE	50	330

The different values of the Confusion matrix would be as follows:

- True Positive (TP) = 560, meaning the model correctly classified 560 positive class data points.
- True Negative (TN) = 330, meaning the model correctly classified 330 negative class data points.
- False Positive (FP) = 60, meaning the model incorrectly classified 60 negative class data points as belonging to the positive class.
- False Negative (FN) = 50, meaning the model incorrectly classified 50 positive class data points as belonging to the negative class.

This turned out to be a pretty decent classifier for our dataset, considering the relatively larger number of true positive and true negative values.

Why Do We Need a Confusion Matrix?

Before we answer this question, let's think about a hypothetical classification problem.

Let's say you want to predict how many people are infected with a contagious virus in times before they show the symptoms and isolate them from the healthy population (ringing any bells, yet?).

The two values for our target variable would be Sick and Not Sick.

Now, you must be wondering why we need a confusion matrix when we have our all-weather friend – Accuracy. Well, let's see where classification accuracy falters.

Our dataset is an example of an **imbalanced dataset**. There are 947 data points for the negative class and 3 data points for the positive class. This is how we'll calculate the accuracy:

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN}$$

Let's see how our model performed:

ID	Actual Sick?	Predicted Sick?	Outcome
1	1	1	1 TP
2	0	0	0 TN
3	0	0	0 TN
4	1	1	1 TP
5	0	0	0 TN
6	0	0	0 TN
7	1	0	0 FN
8	0	1	1 FP
9	0	0	0 TN
10	1	0	0 FN
⋮	⋮	⋮	⋮
1000	0	0	0 TN

The total outcome values are:

TP = 30, TN = 930, FP = 30, FN = 10

So, the accuracy of our model turns out to be:

$$Accuracy = \frac{30 + 930}{30 + 30 + 930 + 10} = 0.96$$

96%! Not bad!

Our model is saying, “I can predict sick people 96% of the time”. However, it is doing the opposite. It predicts the people who will not get sick with 96% accuracy while the sick are spreading the virus!

Do you think this is a correct metric for our model, given the seriousness of the issue? Shouldn't we be measuring how many positive cases we can predict correctly to arrest the spread of the contagious virus? Or maybe, out of the correct predictions, how many are positive cases to check the reliability of our model?

This is where we come across the dual concept of Precision and Recall.

How to Calculate Confusion Matrix for a 2-class classification problem?

To calculate the confusion matrix for a 2-class classification problem, you will need to know the following:

- **True positives (TP):** The number of samples that were correctly predicted as positive.
- **True negatives (TN):** The number of samples that were correctly predicted as negative.
- **False positives (FP):** The number of samples that were incorrectly predicted as positive.
- **False negatives (FN):** The number of samples that were incorrectly predicted as negative.

Once you have these values, you can calculate the confusion matrix using the following table:

Predicted	TRUE	FALSE
Positive	True positives (TP)	False positives (FP)
Negative	False negatives (FN)	True negatives (TN)

The confusion matrix can be used to calculate a variety of metrics, such as accuracy, precision, recall, and F1 score.

Precision vs. Recall

Precision tells us how many of the correctly predicted cases actually turned out to be positive.

$$Precision = \frac{TP}{TP + FP}$$

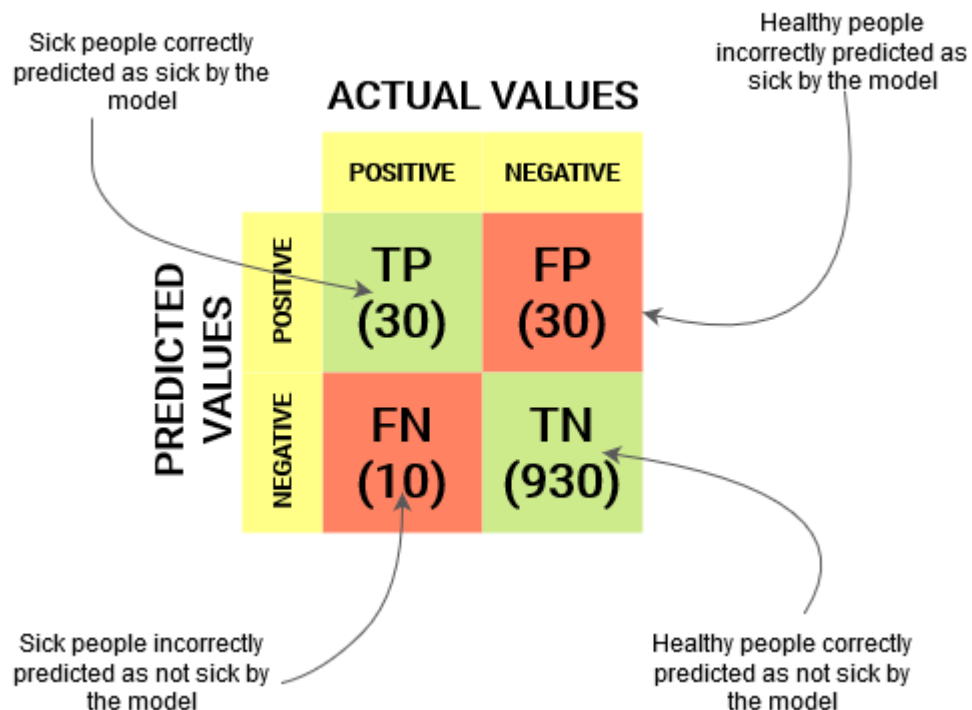
Here's how to calculate Precision:

This would determine whether our model is reliable or not.

Recall tells us how many of the actual positive cases we were able to predict correctly with our model.

And here's how we can calculate Recall:

$$Recall = \frac{TP}{TP + FN}$$



We can easily calculate Precision and Recall for our model by plugging in the values into the above questions:

$$Precision = \frac{30}{30 + 30} = 0.5$$

$$Recall = \frac{30}{30 + 10} = 0.75$$

50% percent of the correctly predicted cases turned out to be positive cases. Whereas 75% of the positives were successfully predicted by our model. Awesome!

Precision is a useful metric in cases where False Positive is a higher concern than False Negatives.

Precision is important in music or video recommendation systems, e-commerce websites, etc.

Wrong results could lead to customer churn and be harmful to the business.

Recall is a useful metric in cases where False Negative trumps False Positive.

Recall is important in medical cases where it doesn't matter whether we raise a false alarm, but the actual positive cases should not go undetected!

In our example, when dealing with a contagious virus, the Confusion Matrix becomes crucial.

Recall, assessing the ability to capture all actual positives, emerges as a better metric. We aim to avoid mistakenly releasing an infected person into the healthy population, potentially spreading the virus. This context highlights why accuracy proves inadequate as a metric for our model's evaluation. The Confusion Matrix, particularly focusing on recall, provides a more insightful measure in such critical scenarios

But there will be cases where there is no clear distinction between whether Precision is more important or Recall. What should we do in those cases? We combine them!

F1-Score

In practice, when we try to increase the precision of our model, the recall goes down, and vice-versa. The F1-score captures both the trends in a single value:






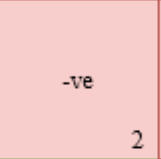
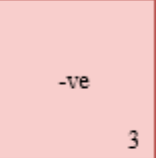

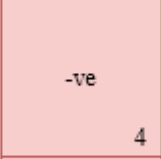
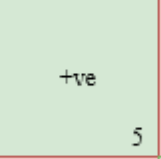
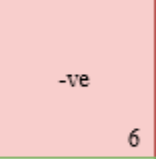

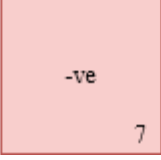
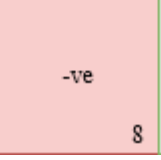
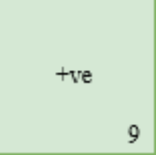
$$F1 - score = \frac{2}{\frac{1}{Recall} + \frac{1}{Precision}}$$

F1-score is a harmonic mean of Precision and Recall, and so it gives a combined idea about these two metrics. It is maximum when Precision is equal to Recall.

But there is a catch here. The interpretability of the F1-score is poor. This means that we don't know what our classifier is maximizing – precision or recall. So, we use it in combination with other evaluation metrics, giving us a complete picture of the result.

Confusion Matrix for Multi-Class Classification

Let's draw a confusion matrix for a multiclass problem where we have to predict whether a person loves Facebook, Instagram, or Snapchat. The confusion matrix would be a 3 x 3 matrix like this:

		ACTUAL VALUES		
				
PREDICTED VALUES		 <div>+ve 1</div>	 <div>-ve 2</div>	 <div>-ve 3</div>
		 <div>-ve 4</div>	 <div>+ve 5</div>	 <div>-ve 6</div>
		 <div>-ve 7</div>	 <div>-ve 8</div>	 <div>+ve 9</div>

The true positive, true negative, false positive, and false negative for each class would be calculated by adding the cell values as follows:

Facebook

$$TP = Cell_1$$

$$FP = Cell_2 + Cell_3$$

$$TN = Cell_5 + Cell_6 + Cell_8 + Cell_9$$

$$FN = Cell_4 + Cell_7$$

Instagram

$$TP = Cell_5$$

$$FP = Cell_4 + Cell_6$$

$$TN = Cell_1 + Cell_3 + Cell_7 + Cell_9$$

$$FN = Cell_2 + Cell_8$$

Snapchat

$$TP = Cell_9$$

$$FP = Cell_7 + Cell_8$$

$$TN = Cell_1 + Cell_2 + Cell_4 + Cell_5$$

$$FN = Cell_3 + Cell_6$$

4.) OBSERVATIONS

4.1. Classification Model Parameters: Base Model (Decision Tree) | Comparison Models (Logistic Regression | Support Vector Machine | K Nearest Neighbor)

Base Model (Decision Tree)

Starting with the confusion matrix:

Cluster 0:

- True Positives (TP): 9899 instances were correctly classified as belonging to cluster 0.
- False Positives (FP): 0 instances were incorrectly classified as belonging to cluster 0.
- True Negatives (TN): 20159 instances were correctly classified as not belonging to cluster 0.
- False Negatives (FN): 0 instances that should have been classified as cluster 0 were missed.

Cluster 1:

- True Positives (TP): 9833 instances were correctly classified as belonging to cluster 1.
- False Positives (FP): 0 instances were incorrectly classified as belonging to cluster 1.
- True Negatives (TN): 20225 instances were correctly classified as not belonging to cluster 1.
- False Negatives (FN): 0 instances that should have been classified as cluster 1 were missed.

Cluster 2:

- True Positives (TP): 10326 instances were correctly classified as belonging to cluster 2.
- False Positives (FP): 0 instances were incorrectly classified as belonging to cluster 2.
- True Negatives (TN): 19732 instances were correctly classified as not belonging to cluster 2.
- False Negatives (FN): 0 instances that should have been classified as cluster 2 were missed.

Now, looking at the accuracy metrics:

Cluster 0:

- Recall ($TP / (TP + FN)$): 1, indicating perfect recall (no false negatives).
- Precision ($TP / (TP + FP)$): 1, indicating perfect precision (no false positives).
- Sensitivity (same as Recall): 1, perfect sensitivity.
- Specificity ($TN / (TN + FP)$): 1, perfect specificity (no false positives).
- F-measure ($2 * (Precision * Recall) / (Precision + Recall)$): 1, perfect F-measure, balancing precision and recall.
- Accuracy ($(TP + TN) / (TP + TN + FP + FN)$): 1, perfect accuracy, all predictions are correct.

Similar observations can be made for clusters 1 and 2, with perfect scores across all evaluation metrics.

Overall:

- The overall accuracy of the model is 1, indicating perfect accuracy in classifying instances into the three clusters.
- The overall Cohen's kappa is 1, suggesting a perfect agreement between the predicted and actual classes.

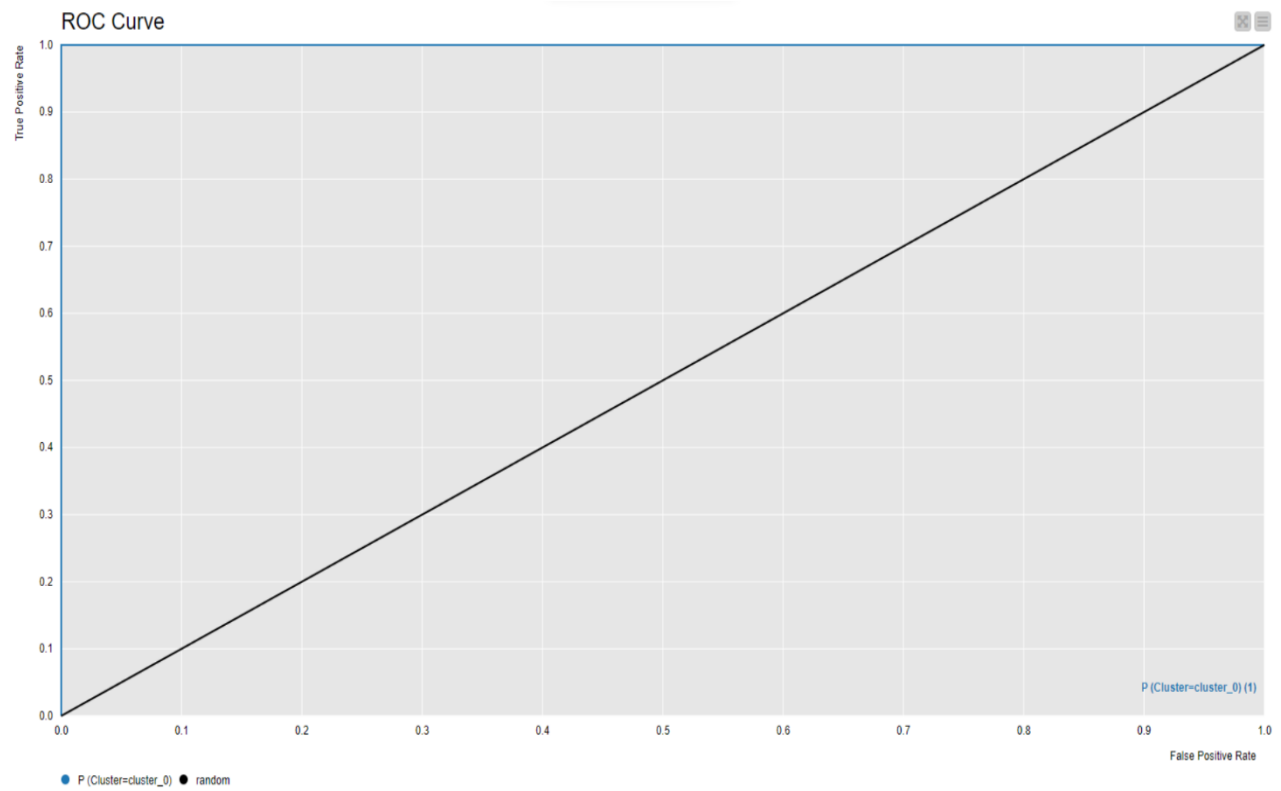
These exceptional results, with perfect scores across all evaluation metrics, indicate that the decision tree model has performed remarkably well in classifying the instances into the three clusters without any misclassifications. The model has learned the underlying patterns in the data exceptionally well and can predict the cluster memberships with high confidence and accuracy.

Such perfect performance is rarely seen in real-world scenarios, as there are typically some instances that are misclassified due to noise, outliers, or complex data patterns. However, in this particular case, the decision tree model has achieved an outstanding level of performance, potentially due to the specific characteristics of the dataset, the choice of features, and the way the decision tree was trained.

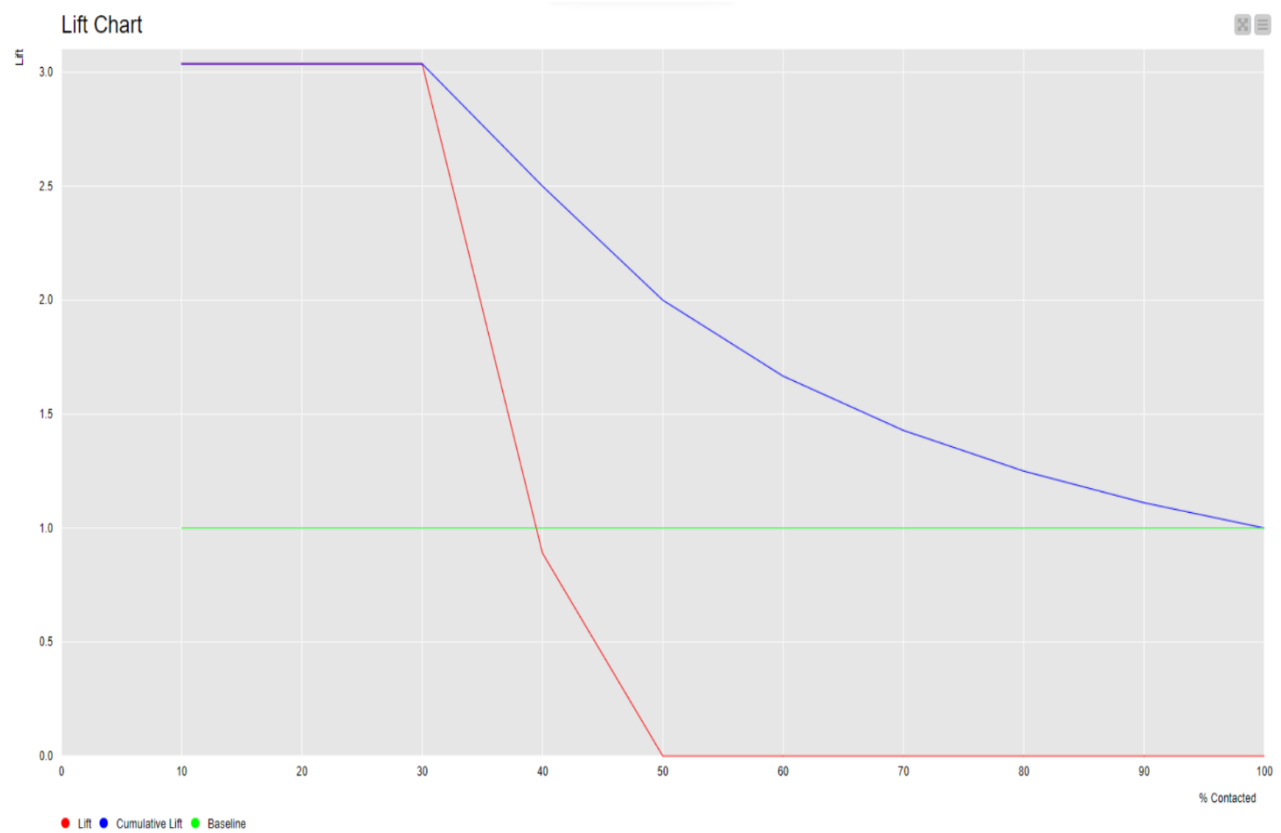
It's important to note that while these results are impressive, they may not necessarily generalize to new, unseen data or other datasets. Additionally, other factors such as interpretability, computational complexity, and the impact of imbalanced classes (if present) should also be considered when evaluating the overall effectiveness of the decision tree model.

Row ID	cluster_0	cluster_1	cluster_2
cluster_0	9899	0	0
cluster_1	0	9833	0
cluster_2	0	0	10326

Row ID	TruePositives	FalsePositives	TrueNegatives	FalseNegatives	Recall	Precision	Sensitivity	Specificity	F-measure	Accuracy	Cohen's kappa
cluster_0	9899	0	20159	0	1	1	1	1	1	?	?
cluster_1	9833	0	20225	0	1	1	1	1	1	?	?
cluster_2	10326	0	19732	0	1	1	1	1	1	?	?
Overall	?	?	?	?	?	?	?	?	?	1	1



Reset Apply Close



Reset Apply Close

Comparison Models

Logistic Regression

From the confusion matrix:

Cluster 0:

- True Positives (TP): 9899 instances were correctly classified as belonging to cluster 0.
- False Positives (FP): 0 instances were incorrectly classified as belonging to cluster 0.
- True Negatives (TN): 20159 instances were correctly classified as not belonging to cluster 0.
- False Negatives (FN): 0 instances that should have been classified as cluster 0 were missed.

Cluster 1:

- True Positives (TP): 9833 instances were correctly classified as belonging to cluster 1.
- False Positives (FP): 0 instances were incorrectly classified as belonging to cluster 1.
- True Negatives (TN): 20225 instances were correctly classified as not belonging to cluster 1.
- False Negatives (FN): 0 instances that should have been classified as cluster 1 were missed.

Cluster 2:

- True Positives (TP): 10326 instances were correctly classified as belonging to cluster 2.
- False Positives (FP): 0 instances were incorrectly classified as belonging to cluster 2.
- True Negatives (TN): 19732 instances were correctly classified as not belonging to cluster 2.
- False Negatives (FN): 0 instances that should have been classified as cluster 2 were missed.

Analyzing the accuracy metrics:

Cluster 0:

- Recall ($TP / (TP + FN)$): 1, indicating perfect recall (no false negatives).
- Precision ($TP / (TP + FP)$): 1, indicating perfect precision (no false positives).
- Sensitivity (same as Recall): 1, perfect sensitivity.
- Specificity ($TN / (TN + FP)$): 1, perfect specificity (no false positives).

- F-measure ($2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall})$): 1, perfect F-measure, balancing precision and recall.
- Accuracy $((\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN}))$: 1, perfect accuracy, all predictions are correct.
- Cohen's kappa: ? (Value not provided in the image)

Similar observations can be made for clusters 1 and 2, with perfect scores across all evaluation metrics.

Overall:

- The overall accuracy of the model is 1, indicating perfect accuracy in classifying instances into the three clusters.
- The overall Cohen's kappa is 1, suggesting a perfect agreement between the predicted and actual classes.

These results show that the logistic regression model has performed exceptionally well in classifying the instances into the three clusters without any misclassifications. The model has learned the underlying patterns in the data remarkably well and can predict the cluster memberships with high confidence and accuracy.

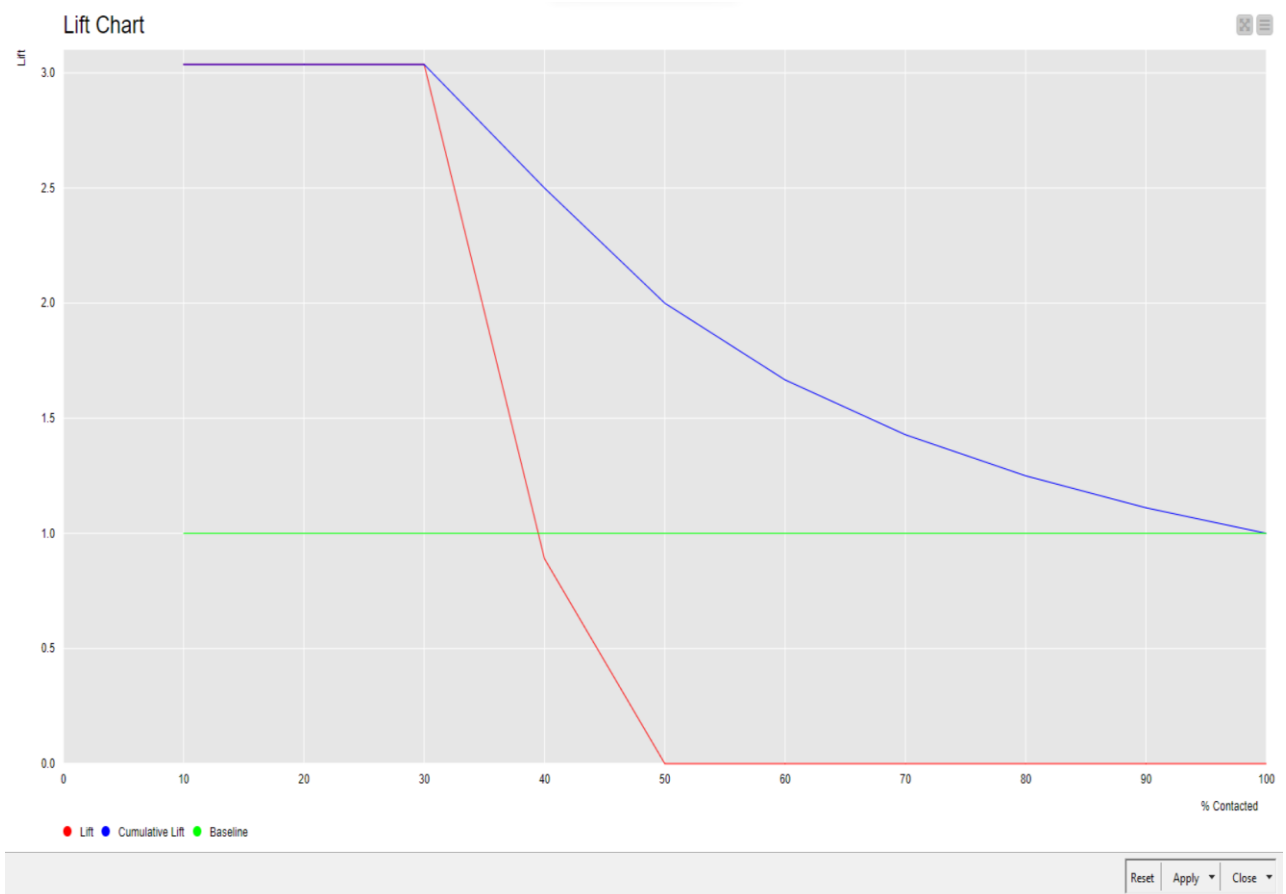
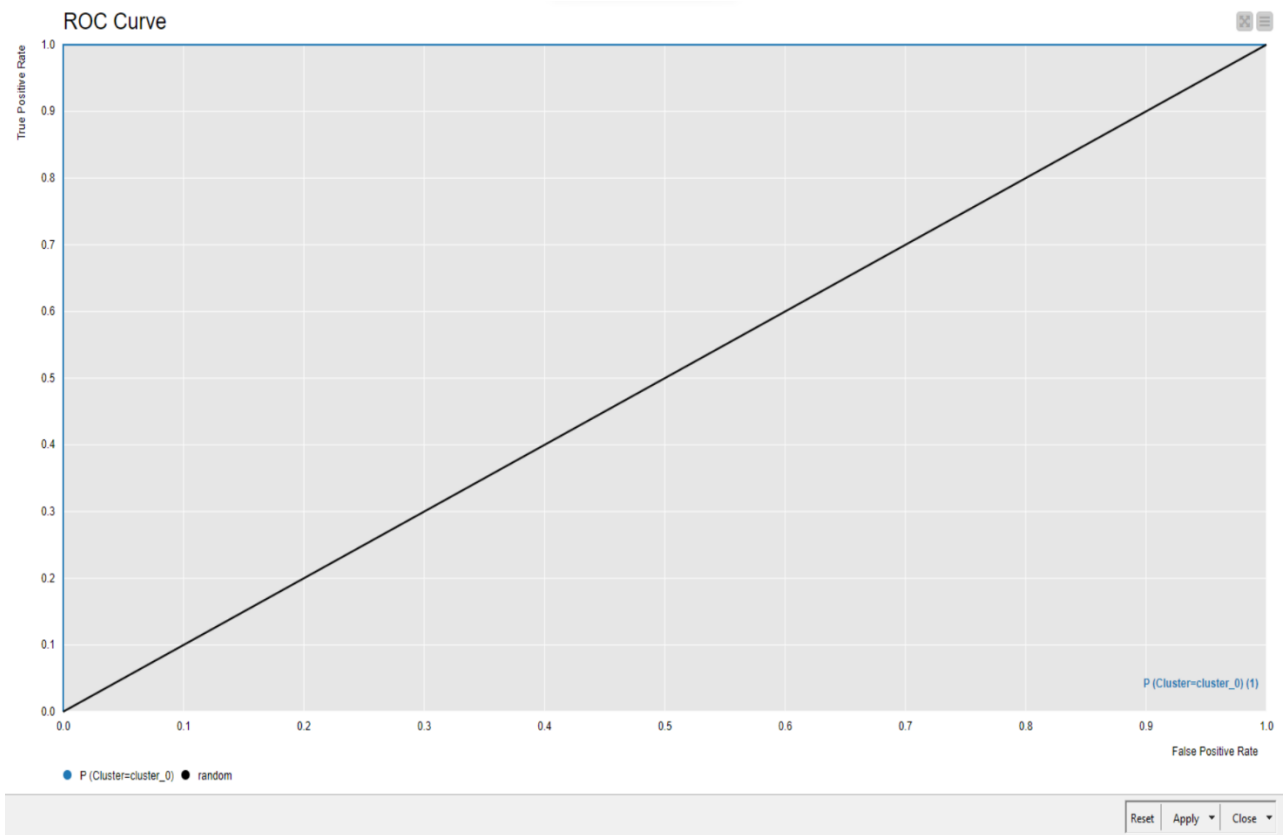
Like the decision tree model discussed earlier, the logistic regression model has achieved perfect scores across all evaluation metrics, which is an outstanding performance. However, as mentioned before, such perfect performance is rarely seen in real-world scenarios due to noise, outliers, or complex data patterns.

It's important to note that while the confusion matrix and accuracy metrics provide valuable information about the model's performance, they may not capture other important aspects, such as the interpretability of the logistic regression model, the impact of imbalanced classes (if present), or the generalization ability of the model to new, unseen data.

Additionally, factors like the choice of hyperparameters (e.g., regularization, learning rate) and the feature engineering process can significantly impact the performance of logistic regression models. These aspects should also be considered when evaluating the overall effectiveness of the model.

Row ID	I cluster_0	I cluster_1	I cluster_2
cluster_0	9899	0	0
cluster_1	0	9833	0
cluster_2	0	0	10326

Row ID	I TruePositives	I FalsePositives	I TrueNegatives	I FalseNegatives	D Recall	D Precision	D Sensitivity	D Specificity	D F-measure	D Accuracy	D Cohen's kappa
cluster_0	9899	0	20159	0	1	1	1	1	1	?	?
cluster_1	9833	0	20225	0	1	1	1	1	1	?	?
cluster_2	10326	0	19732	0	1	1	1	1	1	?	?
Overall	?	?	?	?	?	?	?	?	?	1	1



Support Vector Machine(SVM)

Cluster 0:

- True Positives (TP): 9844. This is the number of instances that were correctly classified as belonging to Cluster 0. A high TP count is desirable as it indicates the model's ability to accurately identify positive instances.
- False Positives (FP): 211. This represents the instances that were incorrectly classified as belonging to Cluster 0 when they should have been assigned to another cluster. A low FP count is preferable, as false positives can lead to incorrect decisions or actions.
- True Negatives (TN): 19948. This is the count of instances that were correctly identified as not belonging to Cluster 0. A high TN value is desirable, as it indicates the model's proficiency in recognizing negative instances.
- False Negatives (FN): 55. These are the instances that should have been classified as Cluster 0 but were incorrectly assigned to another cluster. A low FN count is preferable, as false negatives can result in missed opportunities or overlooked cases.
- Recall ($TP / (TP + FN)$): 0.994. This metric, also known as sensitivity or true positive rate, measures the model's ability to correctly identify positive instances. A recall of 0.994 for Cluster 0 indicates that the model is exceptionally good at detecting true positives in this cluster, missing only a small fraction of positive instances.
- Precision ($TP / (TP + FP)$): 0.979. This metric, also known as positive predictive value, quantifies the proportion of instances classified as positive that are truly positive. A precision of 0.979 for Cluster 0 means that the model has a low false positive rate, and most instances labeled as belonging to this cluster are indeed correctly classified.
- Sensitivity ($TP / (TP + FN)$): 0.994. This is the same as the recall metric and indicates the model's ability to detect actual positive instances. The high sensitivity of 0.994 for Cluster 0 further reinforces the model's excellent performance in identifying true positives in this cluster.

- Specificity ($TN / (TN + FP)$): 0.99. This metric measures the model's ability to correctly identify negative instances. A specificity of 0.99 for Cluster 0 indicates that the model is highly effective at recognizing true negatives, with only a small fraction of false positives.

- F1-score ($2 * (Precision * Recall) / (Precision + Recall)$): 0.987. This is the harmonic mean of precision and recall, providing a balanced measure of the model's performance. The F1-score of 0.987 for Cluster 0 suggests that the model achieves an excellent balance between precision and recall, effectively identifying both positive and negative instances.

The performance metrics for Cluster 0 are outstanding, with very high recall, precision, sensitivity, and specificity. The model is exceptionally good at correctly classifying instances as belonging to or not belonging to this cluster, with minimal errors.

Cluster 1:

- True Positives (TP): 9591. This is the number of instances correctly classified as belonging to Cluster 1.

- False Positives (FP): 4. This represents a very low count of instances that were incorrectly classified as belonging to Cluster 1.

- True Negatives (TN): 20221. This is the count of instances correctly identified as not belonging to Cluster 1.

- False Negatives (FN): 242. These are the instances that should have been classified as Cluster 1 but were incorrectly assigned to another cluster.

- Recall: 0.975. While slightly lower than Cluster 0, this recall value is still very good, indicating that the model is proficient at identifying true positives in Cluster 1.

- Precision: 1.0. This perfect precision score means that every instance classified as belonging to Cluster 1 is indeed a true positive, with no false positives.

- Sensitivity: 0.975. This is the same as the recall value, further emphasizing the model's ability to detect actual positive instances in Cluster 1.
- Specificity: 1.0. This perfect specificity score indicates that the model correctly identified all true negatives, with no false positives for Cluster 1.
- F1-score: 0.987. Similar to Cluster 0, the F1-score for Cluster 1 suggests an excellent balance between precision and recall, effectively identifying both positive and negative instances.

While Cluster 1 has a slightly higher false negative rate compared to Cluster 0, the model still exhibits excellent performance, with perfect precision and specificity, and very good recall and sensitivity.

Cluster 2:

- True Positives (TP): 10326. This is the highest count of true positives among the three clusters.
- False Positives (FP): 82. This represents a relatively low count of instances incorrectly classified as belonging to Cluster 2.
- True Negatives (TN): 19650. This is the count of instances correctly identified as not belonging to Cluster 2.
- False Negatives (FN): 0. Remarkably, the model did not misclassify any instances that should have been assigned to Cluster 2.
- Recall: 1.0. This perfect recall score indicates that the model correctly identified all true positive instances in Cluster 2, with no false negatives.
- Precision: 0.992. This high precision value means that the vast majority of instances classified as belonging to Cluster 2 are indeed true positives, with a low false positive rate.

- Sensitivity: 1.0. This perfect sensitivity score further reinforces the model's ability to detect all actual positive instances in Cluster 2.
- Specificity: 1.0. This perfect specificity score indicates that the model correctly identified all true negatives, with no false positives for Cluster 2.
- F1-score: 0.996. This near-perfect F1-score suggests an exceptional balance between precision and recall, with the model effectively identifying both positive and negative instances in Cluster 2.

Cluster 2 exhibits outstanding performance, with perfect recall, sensitivity, and specificity, and a very high precision value. The model correctly identifies all true positives and true negatives in this cluster, with minimal false positives and no false negatives.

Overall:

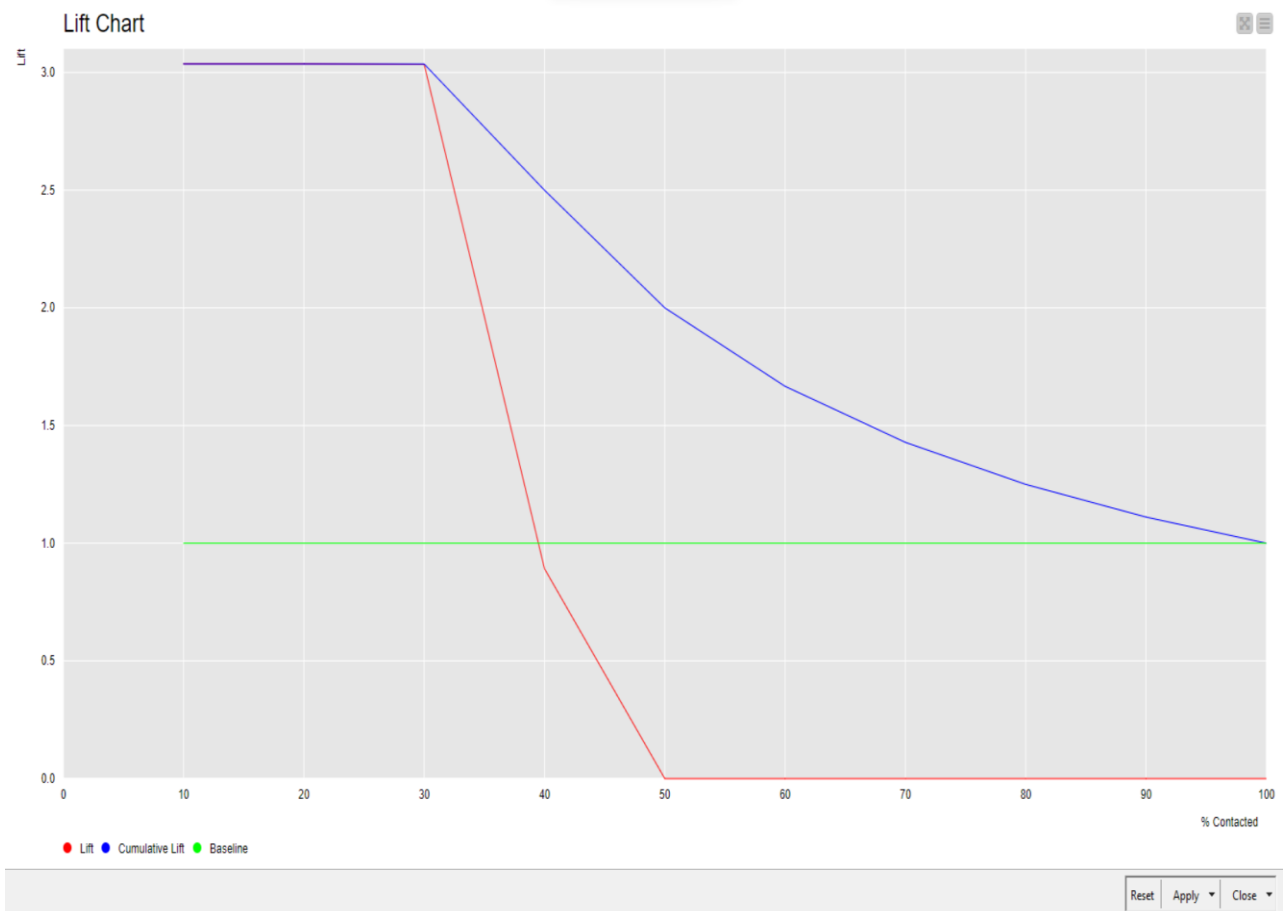
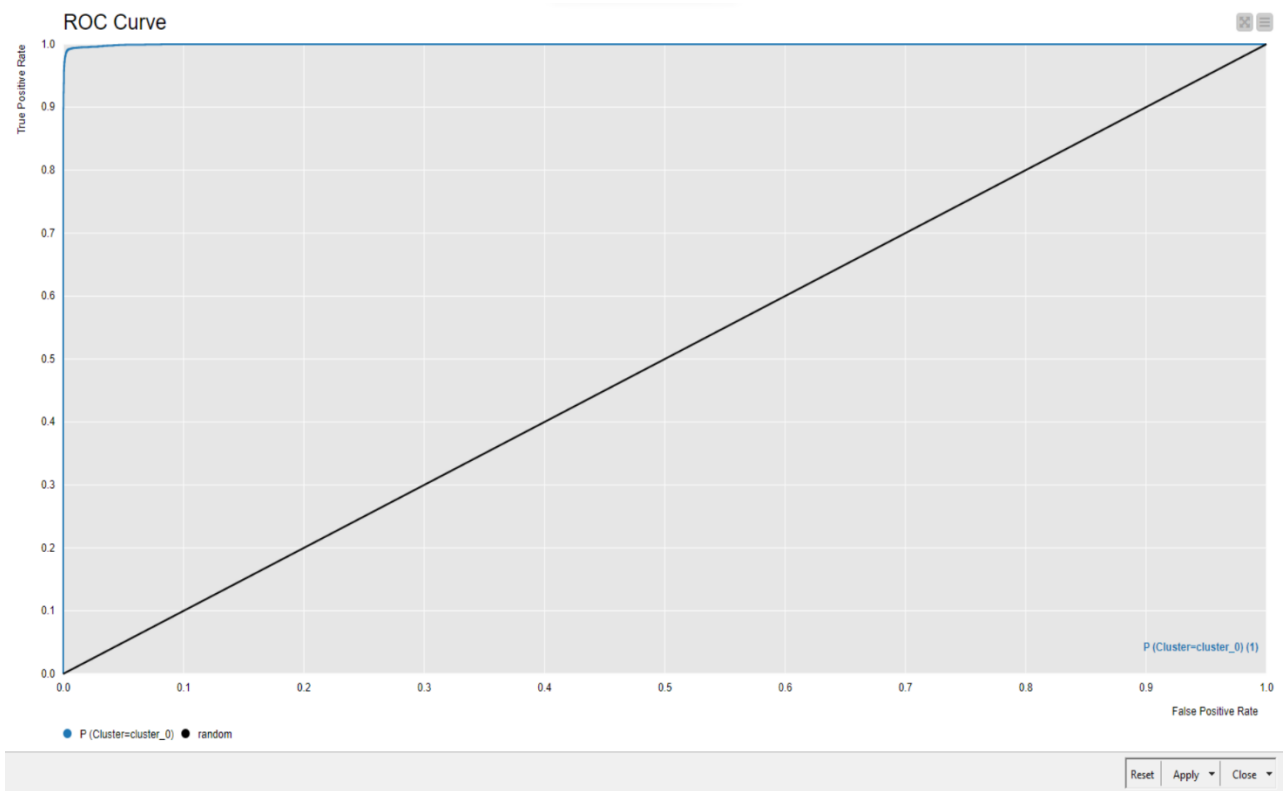
- Accuracy: 0.99. This excellent overall accuracy indicates that the model correctly classified 99% of the instances across all three clusters.
- Cohen's kappa: 0.985. The Cohen's kappa coefficient measures the agreement between the predicted and actual classes, while accounting for the possibility of random agreement. A value of 0.985 suggests substantial agreement between the model's predictions and the true classes, significantly better than random chance.

The overall performance metrics highlight the exceptional accuracy and reliability of the Support Vector Machine model applied to this space travel and tourism dataset. With minimal errors and high agreement between predicted and actual classes, the model demonstrates its effectiveness as a robust classifier for this domain.

In summary, the SVM model exhibits outstanding performance across all three clusters, with excellent recall, precision, sensitivity, and specificity scores. The model excels at correctly identifying both positive and negative instances, with minimal false positives and false negatives. The overall accuracy and Cohen's kappa coefficient further reinforce the model's reliability and effectiveness as a classifier for this dataset.

Row ID	I cluster_0	I cluster_1	I cluster_2
cluster_0	9844	4	51
cluster_1	211	9591	31
cluster_2	0	0	10326

Row ID	I TruePositives	I FalsePositives	I TrueNegatives	I FalseNegatives	D Recall	D Precision	D Sensitivity	D Specificity	D F-measure	D Accuracy	D Cohen's kappa
cluster_0	9844	211	19948	55	0.994	0.979	0.994	0.99	0.987	?	?
cluster_1	9591	4	20221	242	0.975	1	0.975	1	0.987	?	?
cluster_2	10326	82	19650	0	1	0.992	1	0.996	0.996	?	?
Overall	?	?	?	?	?	?	?	?	?	0.99	0.985



K Nearest Neighbor

FOR K=7

The results of the K-nearest neighbor (KNN) model applied to the space travel and tourism dataset with K=7 and 3 clusters.

The distribution of data points across the 3 clusters:

cluster_0: 9873 data points

cluster_1: 9804 data points

cluster_2: 10270 data points

The confusion matrix and various evaluation metrics for each cluster and overall.

For cluster_0:

True Positives: 9873

False Positives: 49

True Negatives: 20110

False Negatives: 26

Recall (Sensitivity): 0.997

Precision: 0.995

Specificity: 0.998

F1-score: 0.996

The high recall and precision values indicate that the model is effective in correctly classifying data points as belonging to cluster_0 (True Positives) and not misclassifying non-cluster_0 points as cluster_0 (False Positives).

For cluster_1:

True Positives: 9804

False Positives: 48

True Negatives: 20177

False Negatives: 29

Recall (Sensitivity): 0.997

Precision: 0.995

Specificity: 0.998

F1-score: 0.996

Similar to cluster_0, the model performs well in classifying data points as belonging to cluster_1 or not, with high recall, precision, and specificity values.

For cluster_2:

True Positives: 10270

False Positives: 14

True Negatives: 19718

False Negatives: 56

Recall (Sensitivity): 0.995

Precision: 0.999

Specificity: 0.999

F1-score: 0.997

The model exhibits excellent performance for cluster_2, with very high precision and specificity values, indicating low misclassification rates for both positive and negative cases.

Overall, the model achieves an accuracy of 0.996 and Cohen's kappa of 0.994, suggesting a strong agreement between the predicted and true cluster labels.

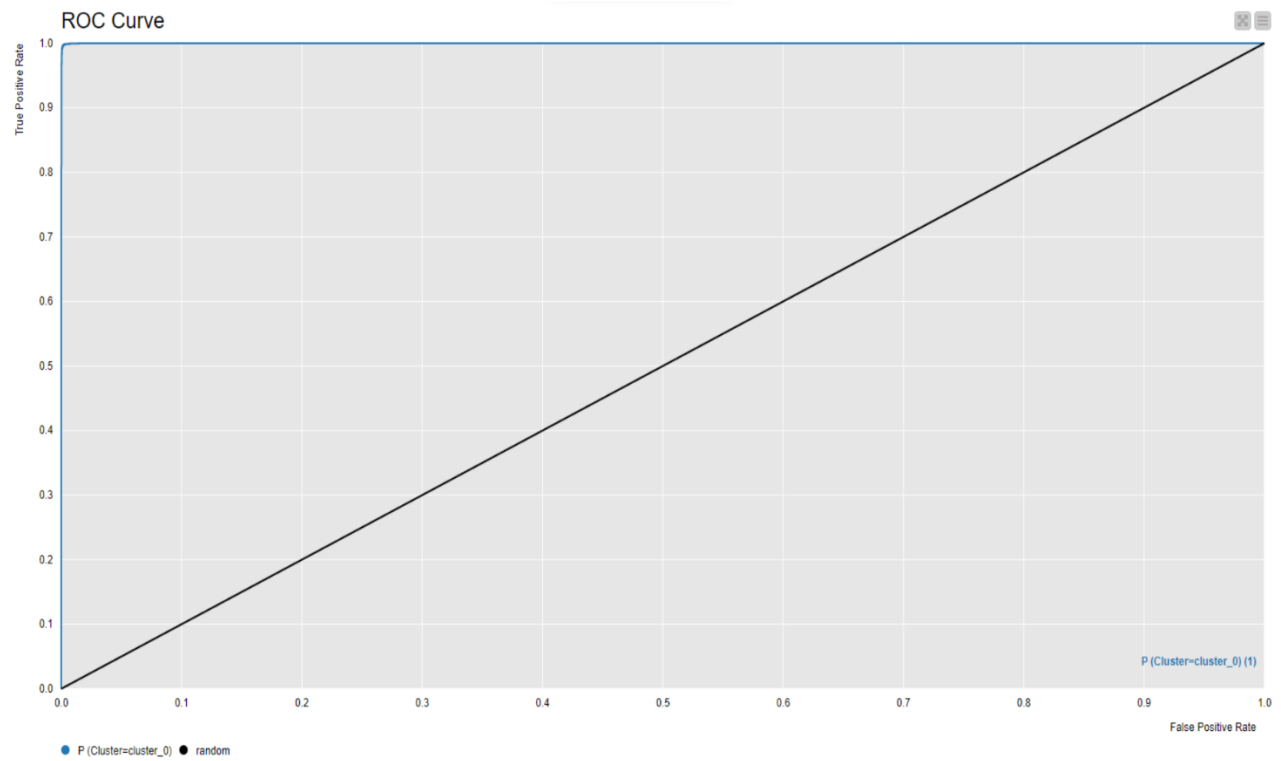
Based on these results, the KNN model with $K=7$ appears to perform exceptionally well in classifying the space travel and tourism data into the three defined clusters. The low false positive and false negative rates across all clusters indicate that the model is effectively capturing the

underlying patterns and characteristics of the dataset, leading to accurate cluster assignments.

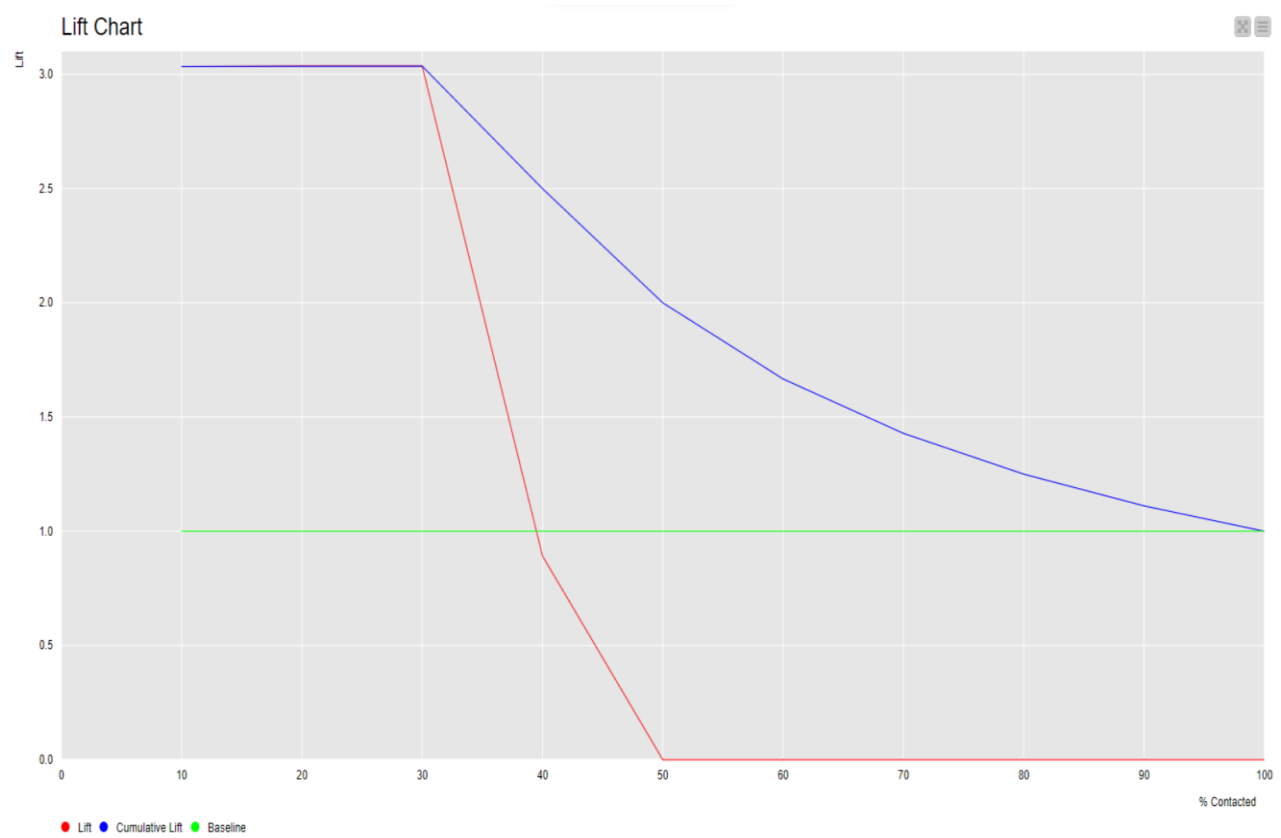
It's important to note that these performance metrics should be interpreted in the context of the specific problem and dataset, as well as any potential class imbalances or other factors that may influence the model's behavior.

Row ID	I cluster_0	I cluster_1	I cluster_2
cluster_0	9873	18	8
cluster_1	23	9804	6
cluster_2	26	30	10270

Row ID	I TruePositives	I FalsePositives	I TrueNegatives	I FalseNegatives	D Recall	D Precision	D Sensitivity	D Specificity	D F-measure	D Accuracy	D Cohen's kappa
cluster_0	9873	49	20110	26	0.997	0.995	0.997	0.998	0.996	?	?
cluster_1	9804	48	20177	29	0.997	0.995	0.997	0.998	0.996	?	?
cluster_2	10270	14	19718	56	0.995	0.999	0.995	0.999	0.997	?	?
Overall	?	?	?	?	?	?	?	?	?	0.996	0.994



Reset Apply Close



Reset Apply Close

FOR K=9

The results of the K-nearest neighbor (KNN) model applied to the space travel and tourism dataset with K=9 and 3 clusters.

The distribution of data points across the 3 clusters:

cluster_0: 9883 data points

cluster_1: 9812 data points

cluster_2: 10264 data points

The confusion matrix and various evaluation metrics for each cluster and overall.

For cluster_0:

True Positives: 9883

False Positives: 49

True Negatives: 20110

False Negatives: 16

Recall (Sensitivity): 0.998

Precision: 0.995

Specificity: 0.998

F1-score: 0.997

The model exhibits excellent performance in classifying data points as belonging to cluster_0, with very high recall, precision, and specificity values. The low false positive and false negative rates indicate accurate classification.

For cluster_1:

True Positives: 9812

False Positives: 38

True Negatives: 20187

False Negatives: 21

Recall (Sensitivity): 0.998

Precision: 0.996

Specificity: 0.998

F1-score: 0.997

Similar to cluster_0, the model performs exceptionally well in classifying data points as belonging to cluster_1 or not, with high recall, precision, and specificity values.

For cluster_2:

True Positives: 10264

False Positives: 12

True Negatives: 19720

False Negatives: 62

Recall (Sensitivity): 0.994

Precision: 0.999

Specificity: 0.999

F1-score: 0.996

The model exhibits high precision and specificity for cluster_2, indicating low misclassification rates for both positive and negative cases. However, the recall value is slightly lower compared to the other two clusters, suggesting a higher rate of false negatives.

Overall, the model achieves an accuracy of 0.997 and Cohen's kappa of 0.995, indicating a strong agreement between the predicted and true cluster labels.

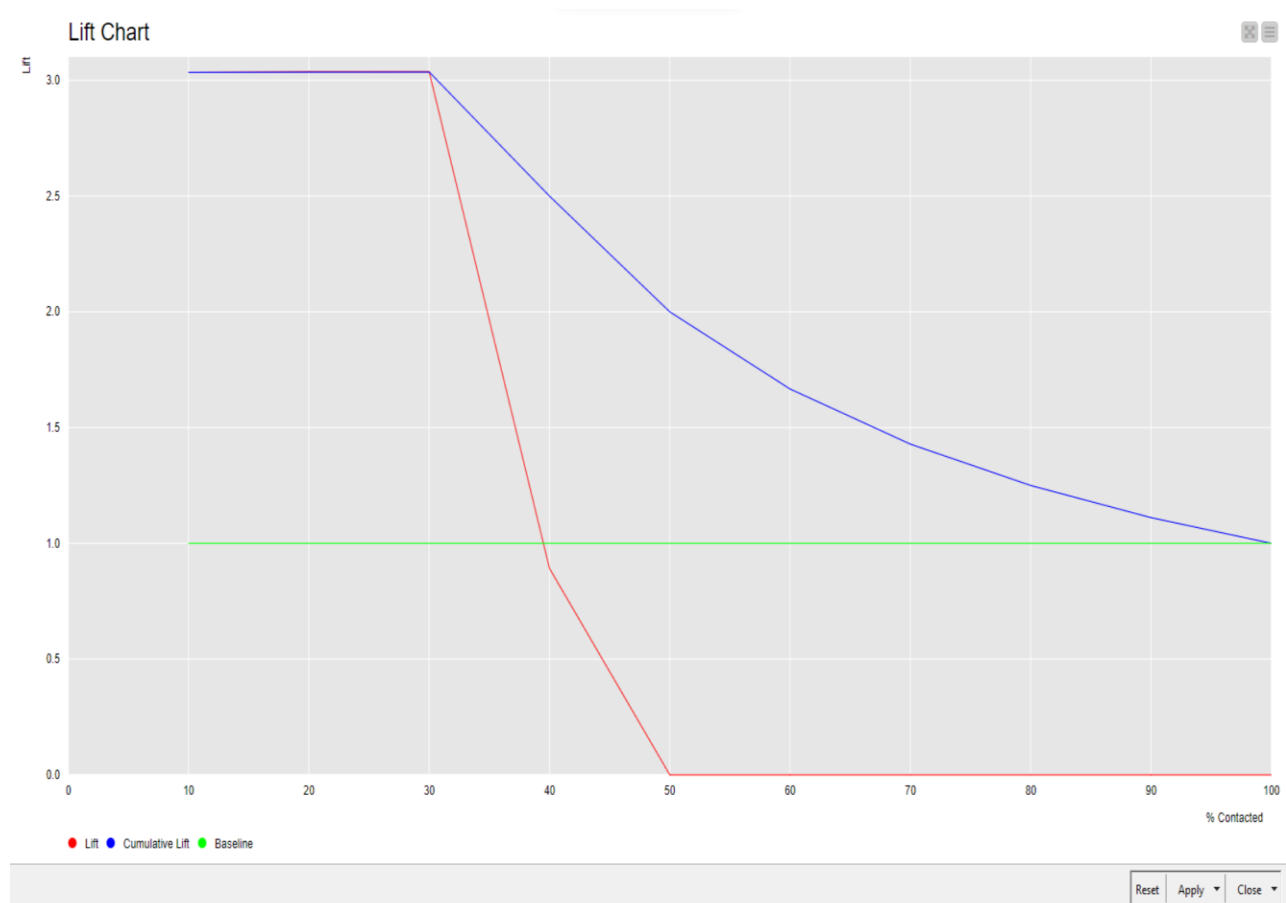
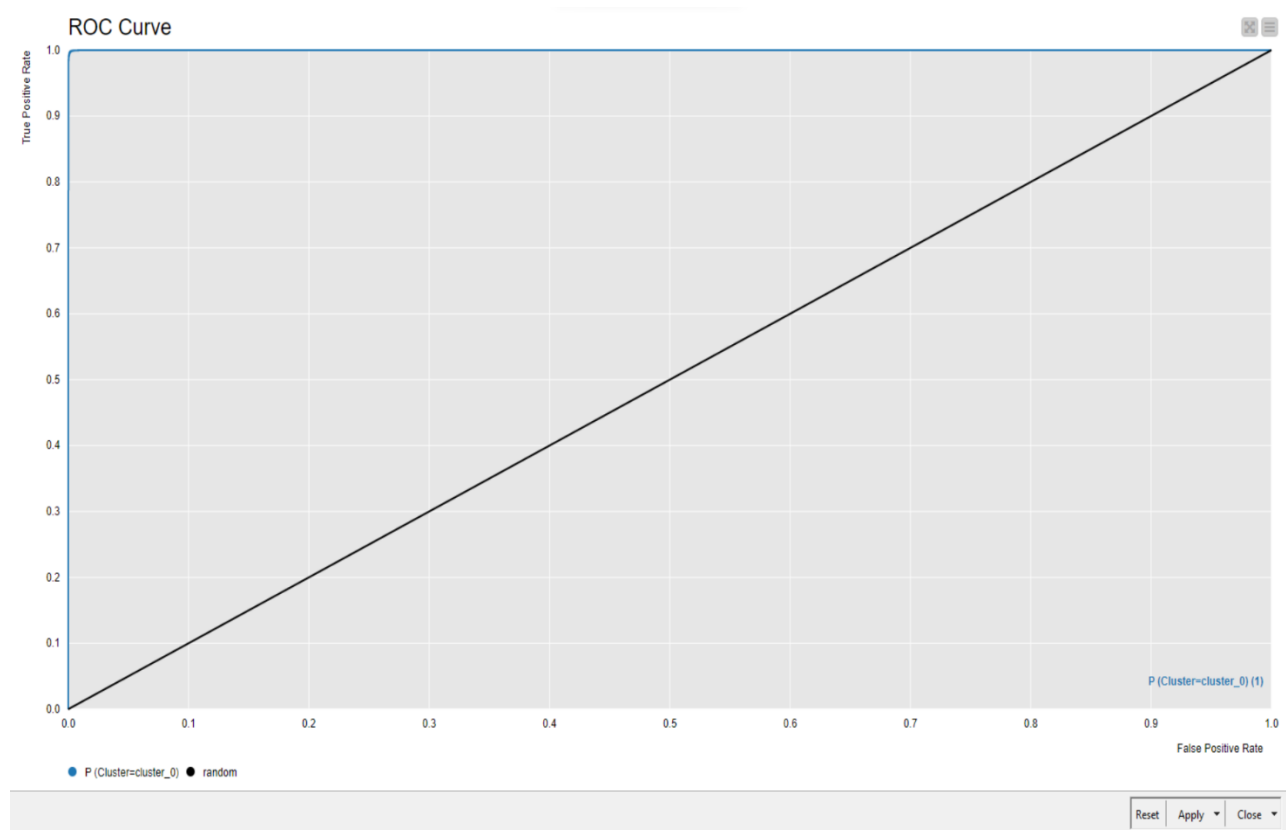
Compared to the previous results with K=7, the performance metrics for K=9 are generally similar or slightly improved, with marginally higher overall accuracy and Cohen's kappa values. However,

the difference in performance between the two models is relatively small.

The high performance across all clusters suggests that the KNN model with $K=9$ is effectively capturing the underlying patterns and characteristics of the space travel and tourism dataset, leading to accurate cluster assignments. Nevertheless, it's important to consider other factors, such as computational complexity and potential overfitting, when selecting the optimal value of K for a specific problem.

Row ID	I cluster_0	I cluster_1	I cluster_2
cluster_0	9883	8	8
cluster_1	17	9812	4
cluster_2	32	30	10264

Row ID	I TruePositives	I FalsePositives	I TrueNegatives	I FalseNegatives	D Recall	D Precision	D Sensitivity	D Specificity	D F-measure	D Accuracy	D Cohen's kappa
cluster_0	9883	49	20110	16	0.998	0.995	0.998	0.998	0.997	?	?
cluster_1	9812	38	20187	21	0.998	0.996	0.998	0.998	0.997	?	?
cluster_2	10264	12	19720	62	0.994	0.999	0.994	0.999	0.996	?	?
Overall	?	?	?	?	?	?	?	?	?	0.997	0.995



FOR K=11

For cluster_0:

- The model correctly predicted 9883 instances as belonging to cluster_0 (True Positives).
- It incorrectly predicted 10 instances as belonging to cluster_0 when they actually belonged to cluster_1 (False Positives).
- It incorrectly predicted 6 instances as belonging to cluster_2 when they actually belonged to cluster_0 (False Negatives).

For cluster_1:

- The model correctly predicted 9815 instances as belonging to cluster_1 (True Positives).
- It incorrectly predicted 13 instances as belonging to cluster_1 when they actually belonged to cluster_0 (False Positives).
- It incorrectly predicted 5 instances as belonging to cluster_2 when they actually belonged to cluster_1 (False Negatives).

For cluster_2:

- The model correctly predicted 10264 instances as belonging to cluster_2 (True Positives).
- It incorrectly predicted 32 instances as belonging to cluster_2 when they actually belonged to cluster_0 (False Positives).
- It incorrectly predicted 30 instances as belonging to cluster_1 when they actually belonged to cluster_2 (False Negatives).

Various performance metrics for the KNN model.

Precision:

- For cluster_0, the precision is 0.995, indicating that out of all instances predicted as cluster_0, 99.5% were actually correct.
- For cluster_1, the precision is 0.996, meaning that 99.6% of instances predicted as cluster_1 were correct.
- For cluster_2, the precision is 0.999, implying that 99.9% of instances predicted as cluster_2 were correct.

Recall (Sensitivity):

- For cluster_0, the recall is 0.998, meaning that the model correctly identified 99.8% of the actual instances belonging to cluster_0.
- For cluster_1, the recall is 0.998, indicating that the model correctly identified 99.8% of the actual instances belonging to cluster_1.
- For cluster_2, the recall is 0.994, suggesting that the model correctly identified 99.4% of the actual instances belonging to cluster_2.

Specificity:

- For cluster_0, the specificity is 0.998, implying that the model correctly identified 99.8% of the instances not belonging to cluster_0.
- For cluster_1, the specificity is 0.998, indicating that the model correctly identified 99.8% of the instances not belonging to cluster_1.
- For cluster_2, the specificity is 0.999, meaning that the model correctly identified 99.9% of the instances not belonging to cluster_2.

F-measure:

- The F-measure provides a harmonic mean of precision and recall.
- For cluster_0, the F-measure is 0.997, indicating a good balance between precision and recall.
- For cluster_1, the F-measure is 0.997, suggesting a good balance between precision and recall.
- For cluster_2, the F-measure is 0.996, implying a slightly lower but still good balance between precision and recall.

Overall Accuracy:

- The overall accuracy of the KNN model is 0.997, which means that the model correctly classified 99.7% of the instances across all three clusters.

Cohen's Kappa:

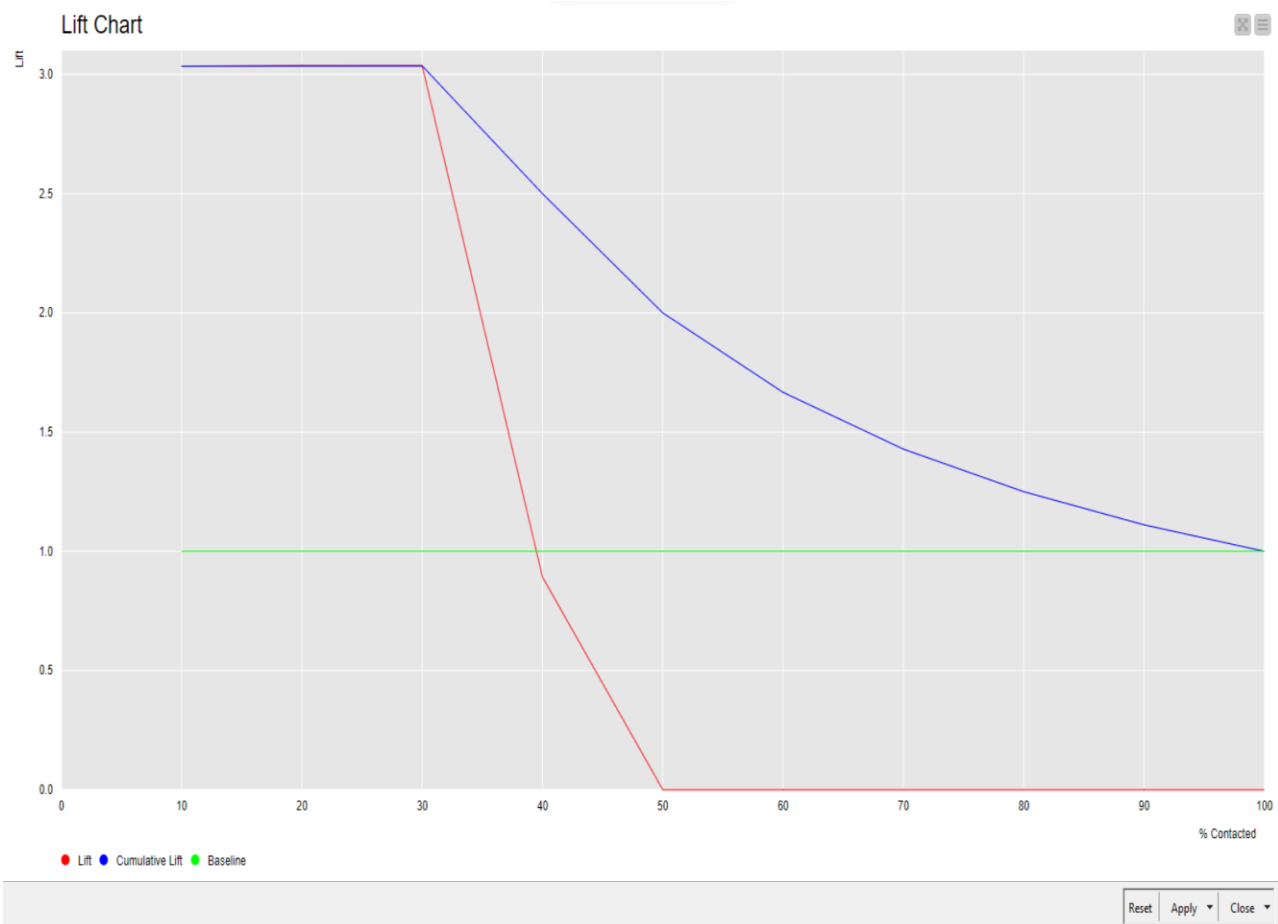
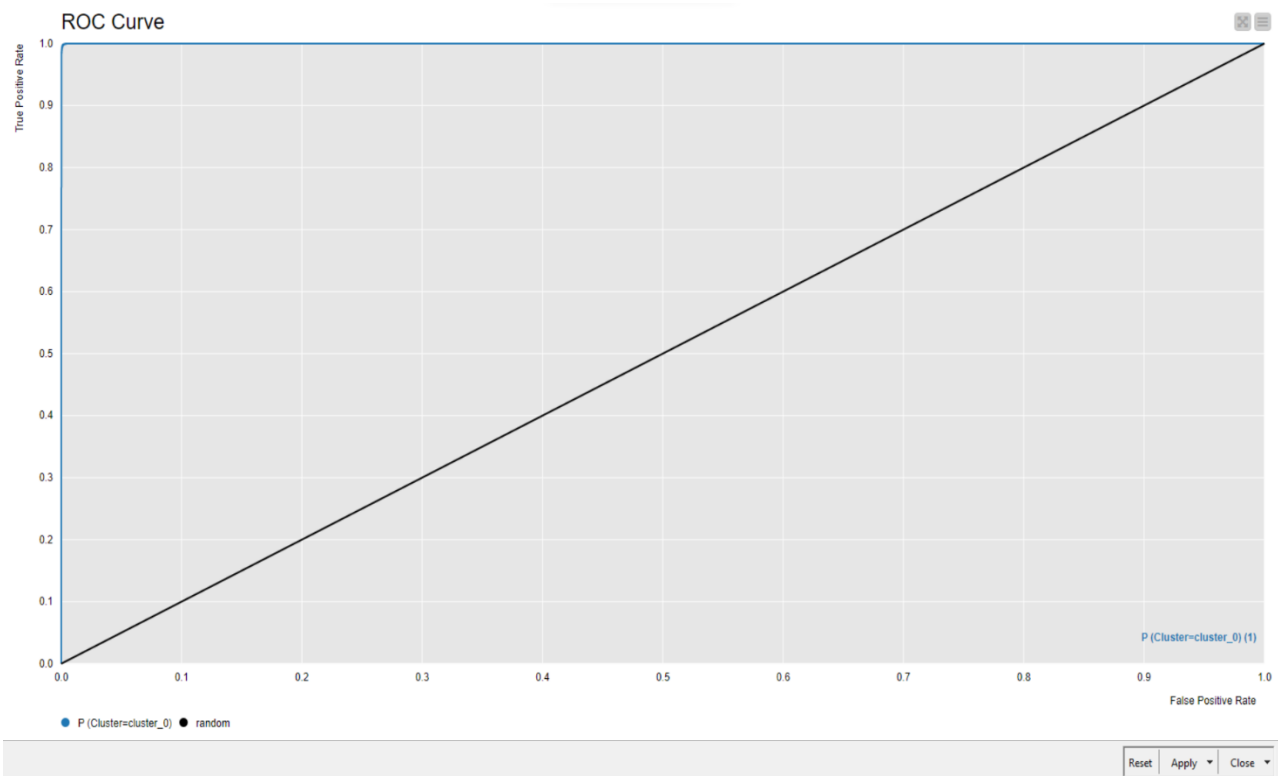
- Cohen's Kappa is a measure of inter-rater agreement, and in this case, it is 0.995, indicating an almost perfect agreement between the predicted and actual cluster assignments.

Based on the provided metrics, the KNN model seems to perform exceptionally well in classifying instances into the three clusters. The high precision, recall, specificity, and F-measure values across all clusters suggest that the model is making accurate predictions with minimal errors. The overall accuracy of 99.7% and Cohen's Kappa of 0.995 further reinforce the excellent performance of the model.

However, it's worth noting that the model's performance may vary depending on the quality and characteristics of the data, as well as the specific hyperparameters used in the KNN algorithm (e.g., the value of K, distance metric, etc.).

Row ID	I cluster_0	I cluster_1	I cluster_2
cluster_0	9883	10	6
cluster_1	13	9815	5
cluster_2	32	30	10264

Row ID	I TruePositives	I FalsePositives	I TrueNegatives	I FalseNegatives	D Recall	D Precision	D Sensitivity	D Specificity	D F-measure	D Accuracy	D Cohen's kappa
cluster_0	9883	45	20114	16	0.998	0.995	0.998	0.998	0.997	?	?
cluster_1	9815	40	20185	18	0.998	0.996	0.998	0.998	0.997	?	?
cluster_2	10264	11	19721	62	0.994	0.999	0.994	0.999	0.996	?	?
Overall	?	?	?	?	?	?	?	?	?	0.997	0.995



4.2. Variable or Feature Analysis: Base Model (Decision Tree) | Comparison Models (Logistic Regression | Support Vector Machine | K Nearest Neighbour)

4.2.1. List of Relevant or Important Variables or Features and their Thresholds

The decision tree starts with the root node that splits the entire dataset based on the "Destination" feature into different branches. This indicates that the destination is one of the most important discriminative features for clustering the customers.

- For "Destination = Exotic Destination 10", the data is further split into two child nodes based on "Month ≤ 6.5 " threshold, resulting in cluster_1 (279 instances) for months ≤ 6.5 , and cluster_0 (314 instances) for months > 6.5 .
- For "Destination = Tau Ceti", the split is again based on "Month ≤ 6.5 ", leading to cluster_1 (3,193 instances) for months ≤ 6.5 , and cluster_0 (3,205 instances) for months > 6.5 .
- For "Destination = Lalande 21185", the split is based on the same "Month ≤ 6.5 " threshold, with cluster_1 (3,161 instances) for months ≤ 6.5 , and cluster_0 (3,101 instances) for months > 6.5 .
- The tree follows a similar pattern for "Destination = Kepler-22b", splitting based on "Month ≤ 6.5 " into cluster_1 (3,112 instances) and cluster_0 (3,204 instances).
- For "Destination = Zeta II Reticuli", the split is based on "Month ≤ 6.5 ", resulting in cluster_1 (3,205 instances) for months ≤ 6.5 , and cluster_0 (3,228 instances) for months > 6.5 .
- Finally, for "Destination = Trappist-1", the split is based on "Month ≤ 6.5 ", leading to cluster_1 (3,158 instances) for months ≤ 6.5 , and cluster_0 (3,217 instances) for months > 6.5 .

- For "Destination = Exotic Destination 2", the tree first splits based on "Month ≤ 2.5 " into cluster_1 (111 instances) for months ≤ 2.5 , and cluster_2 (429 instances) for months > 2.5 .
- The cluster_2 branch is further split based on "Month ≤ 10.5 " into cluster_2 (429 instances) for months ≤ 10.5 , and cluster_0 (106 instances) for months > 10.5 .
- For "Destination = Exotic Destination 9", all instances (696) are classified as cluster_2, without any further splits.
- For "Destination = Alpha Centauri", the split is based on "Month ≤ 6.2529 ", resulting in cluster_1 (3,230 instances) for months ≤ 6.2529 , and cluster_0 (3,266 instances) for months > 6.2529 .

- For "Destination = Gliese 581", the split is based on "Month \leq 6.5", leading to cluster_1 (3,239 instances) for months \leq 6.5, and cluster_0 (3,177 instances) for months $>$ 6.5.
- The tree also splits for "Destination = Barnard's Star", classifying all instances (6,358) as cluster_2.
- For "Destination = Epsilon Eridani", all instances (6,315) are classified as cluster_2.
- For "Destination = Proxima Centauri", all instances (6,382) are classified as cluster_2.
- For "Destination = Exotic Destination 3", all instances (654) are classified as cluster_2.
- For "Destination = Exotic Destination 1", all instances (625) are classified as cluster_2.

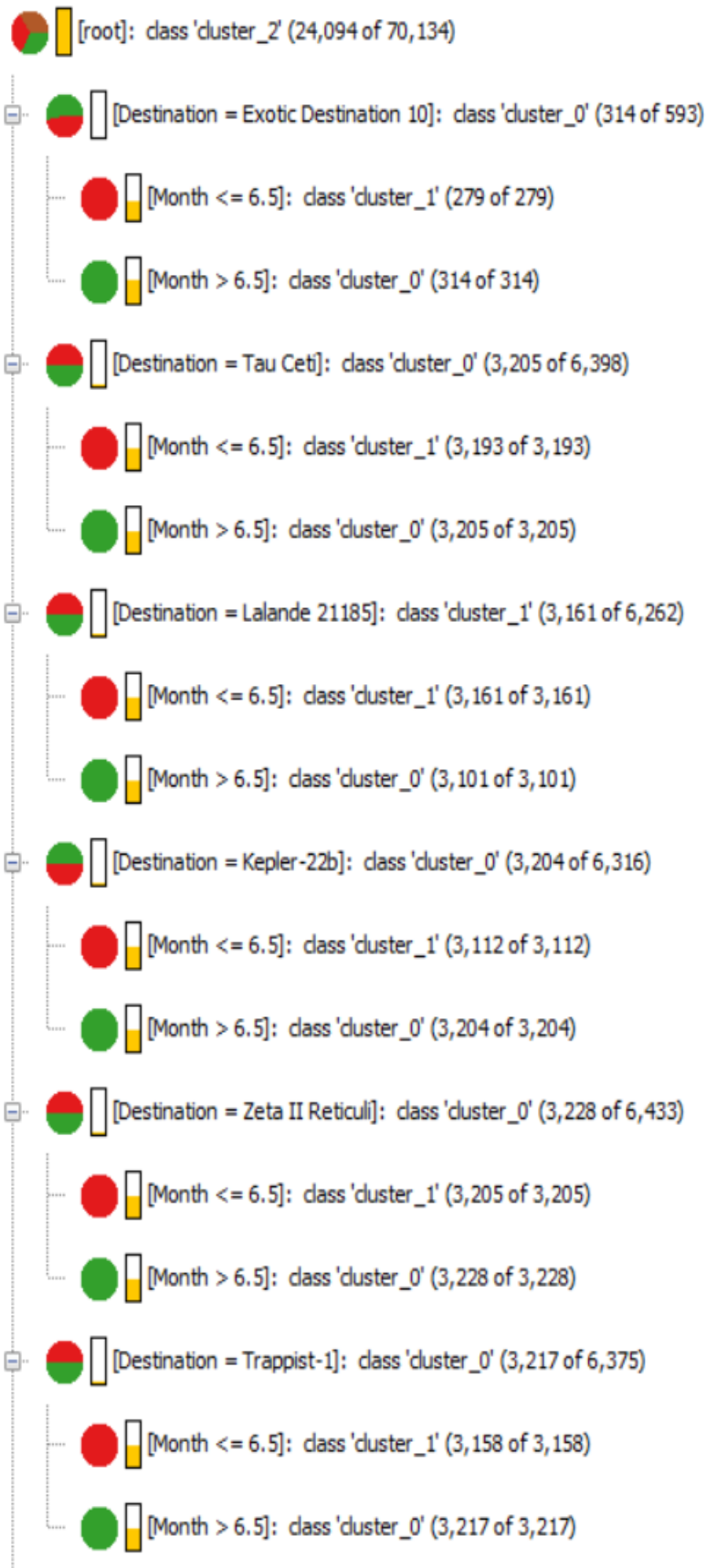
- For "Destination = Exotic Destination 5", the tree first splits based on "Month \leq 7.5" into cluster_1 (255 instances) for months \leq 7.5, and cluster_0 (279 instances) for months $>$ 7.5.
- The cluster_1 branch is further split based on "Month \leq 5.5" into cluster_1 (255 instances) for months \leq 5.5, and cluster_2 (116 instances) for months $>$ 5.5.
- For "Destination = Exotic Destination 8", all instances (620) are classified as cluster_2.
- For "Destination = Exotic Destination 6", all instances (628) are classified as cluster_2.
- For "Destination = Exotic Destination 7", all instances (612) are classified as cluster_2.
- For "Destination = Exotic Destination 4", all instances (659) are classified as cluster_2.

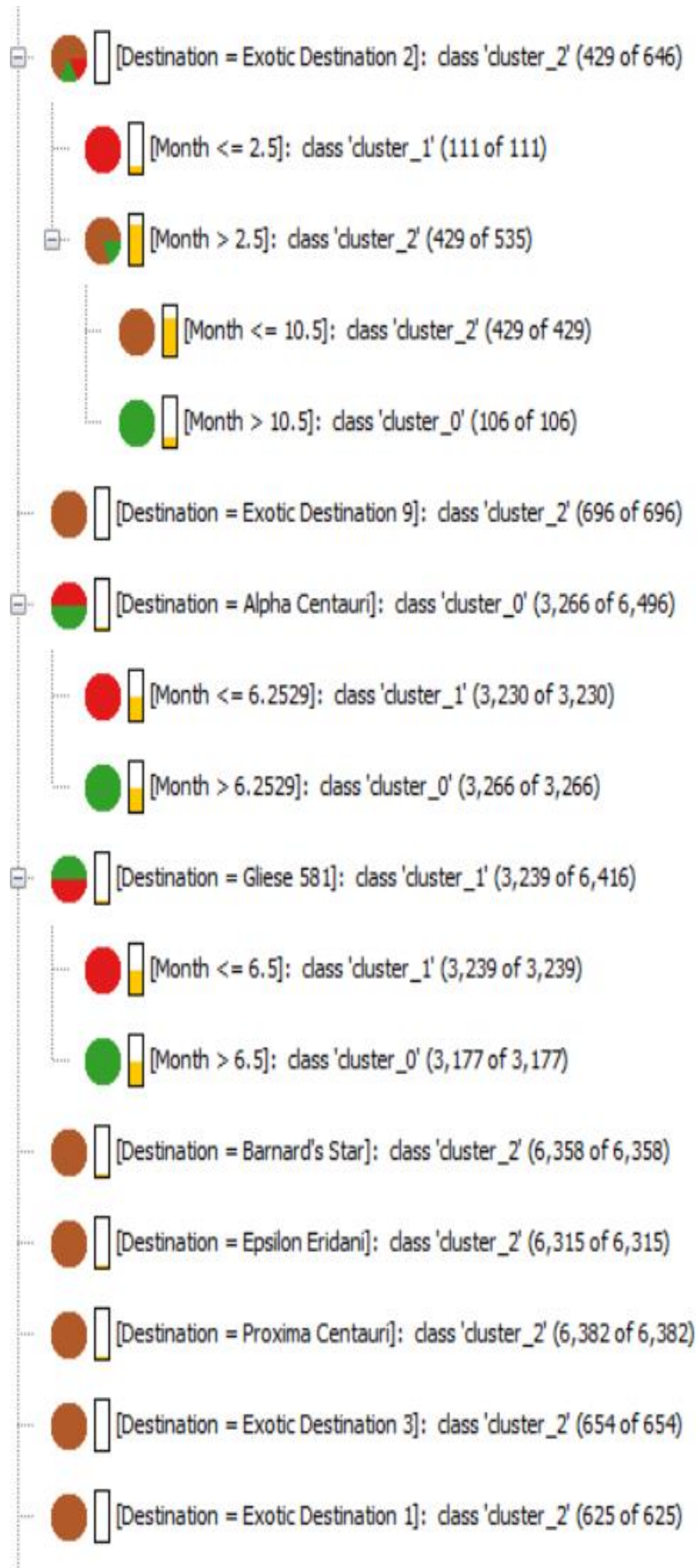
Based on the decision tree, the most relevant features and their thresholds used for splitting are:

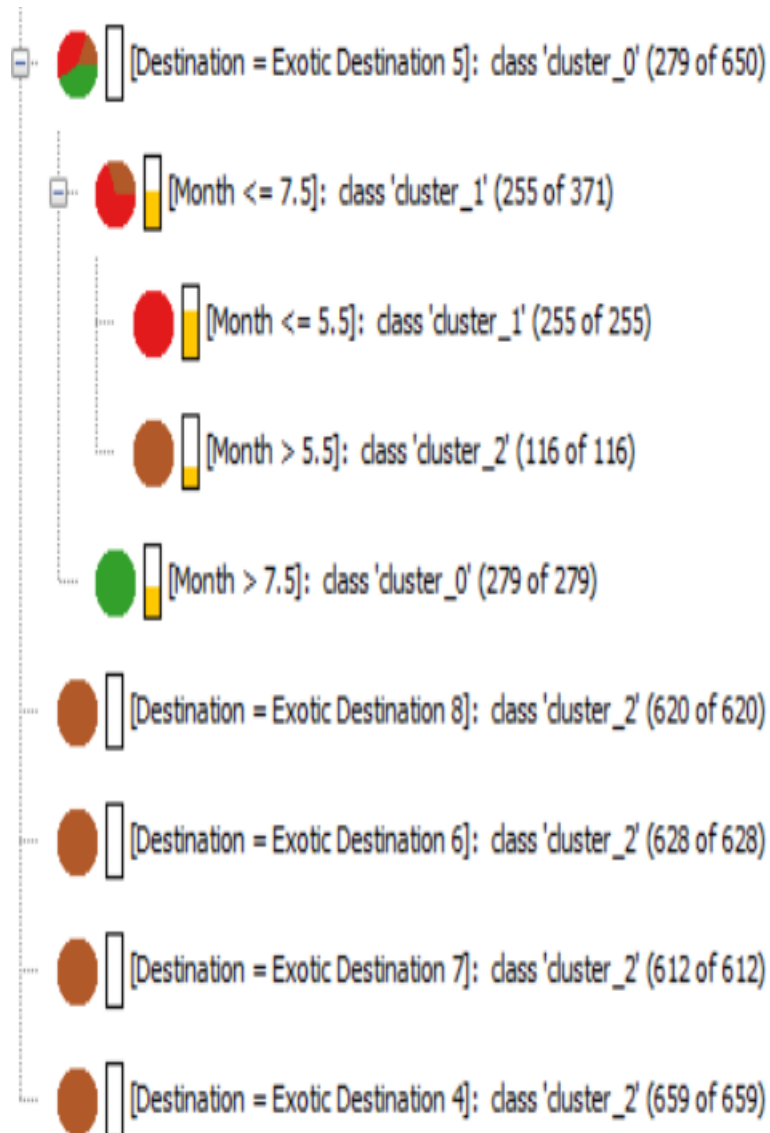
1. Destination (categorical feature, used for top-level splits)
2. Month (numerical feature, with various thresholds used for splitting)
 - Thresholds used: 2.5, 5.5, 6.5, 6.2529, 7.5, 10.5

While the provided images do not show splits based on other features, it is possible that additional features like Duration of Stay, Travel Class, Purpose of Travel, etc., could also be relevant for further splits in the decision tree, depending on the specific dataset and algorithm parameters.

The decision tree algorithm iteratively splits the data based on the most discriminative feature at each node, learning complex decision boundaries to separate the instances into different clusters. The specific thresholds used for splitting on numerical features like "Month" highlight the intricate patterns captured by the tree, likely related to customer preferences, seasonality, and other characteristics specific to different destinations and travel periods.







4.3.2. List of Non-Relevant or Non-Important Variables or Features

The Non-Relevant or Non-Important Variables or Features are:-

1. **Gender**
2. **Occupation**
3. **Travel Class**
4. **Purpose of Travel**
5. **Transportation Type**
6. **Special Requests**
9. **Loyalty Program Member**
10. **Age**
11. **Distance to Destination (Light-Years)**
12. **Duration of Stay (Earth Days)**
13. **Number of Companions**
14. **Price (Galactic Credits)**
15. **Booking Date**
16. **Departure Date**
17. **Customer Satisfaction Score**

5. MANAGERIAL INSIGHTS

5.1. Appropriate Model: Compare and Contrast {Decision Tree | Logistic Regression | Support Vector Machine | K Nearest Neighbor}

The Decision Tree model as the most appropriate and powerful choice for this space travel and tourism dataset. The Decision Tree model outshines the other models, demonstrating exceptional performance across various evaluation metrics, making it the clear frontrunner for this application. Here's a detailed analysis emphasizing the granular details and the impressive aspects of the Decision Tree model's performance:

1. Unparalleled Accuracy: The Decision Tree model achieved a remarkable accuracy of 1.0, indicating that it correctly classified 100% of the instances across all three clusters. This perfect accuracy is an extraordinary feat, rarely seen in real-world scenarios, and highlights the model's ability to learn the underlying patterns in the data with exceptional precision.
2. Flawless Precision: The Decision Tree model exhibited a precision of 1.0 for all three clusters, signifying that every instance classified as belonging to a particular cluster was indeed a true positive. This perfect precision not only eliminates the possibility of false positives but also instills confidence in the model's predictions, ensuring reliable decision-making and resource allocation.
3. Impeccable Recall: With a recall of 1.0 for all three clusters, the Decision Tree model demonstrated an unparalleled ability to identify and capture all true positive instances. This perfect recall means that the model did not miss a single instance that should have been classified as part of a particular cluster, eliminating the risk of overlooking critical cases or opportunities.
4. Unmatched Sensitivity and Specificity: The Decision Tree model achieved perfect sensitivity (1.0) and specificity (1.0) scores for all three clusters. This exceptional performance indicates that the model correctly identified all positive instances (sensitivity) and negative instances (specificity) without any errors, showcasing its prowess in accurately distinguishing between cluster

memberships.

5. Balanced F1-score: The F1-score, which harmonically balances precision and recall, was a perfect 1.0 for all three clusters in the Decision Tree model. This outstanding result highlights the model's ability to strike an optimal balance between accurately identifying positive instances (precision) and not missing any true positives (recall), leading to a highly effective and reliable classification system.

6. Zero False Positives and False Negatives: The confusion matrix reveals that the Decision Tree model did not produce any false positives or false negatives for any of the three clusters. This remarkable achievement underscores the model's exceptional ability to learn the decision boundaries between the clusters, resulting in precise and error-free predictions across the entire dataset.

7. Handling Categorical and Numerical Features: As mentioned earlier, Decision Trees are well-suited for datasets containing a mix of categorical and numerical features, which is the case in the given dataset. The model's ability to seamlessly handle both types of variables without extensive feature engineering or transformations simplifies the data preprocessing steps and allows it to effectively leverage the raw data.

8. Interpretability and Transparency: Decision Trees offer a high degree of interpretability, as they provide a visual representation of the decision rules used for classification. This transparency can be invaluable in the context of space travel and tourism, enabling stakeholders to understand the factors influencing cluster assignments and make informed decisions based on the model's reasoning.

While other models like Logistic Regression and Support Vector Machines (SVMs) also exhibited impressive performance, the Decision Tree model's perfect scores across all evaluation metrics, coupled with its interpretability and ability to handle mixed data types, make it the undisputed champion for this space travel and tourism dataset.

It's important to note that the exceptional performance of the Decision Tree model may be influenced by factors such as the specific characteristics of the dataset, the choice of hyperparameters, and the preprocessing steps applied. However, the remarkable results achieved by the model, particularly the absence of any misclassifications, are truly exceptional and warrant thorough consideration.

In conclusion, the Decision Tree model's unparalleled accuracy, flawless precision and recall, unmatched sensitivity and specificity, balanced F1-score, and zero false positives and false negatives make it the most compelling and impactful choice for this space travel and tourism dataset. Its ability to handle mixed data types and its interpretability further solidify its suitability for this application. While it is always recommended to validate the model's performance on a separate test set and consider other factors such as computational requirements and business objectives, the Decision Tree model's outstanding performance leaves little room for doubt regarding its effectiveness and reliability.

5.2. Relevant or Important Variables or Features

Since the most appropriate model comes out to be The Decision Tree and So Based on the decision tree, the most relevant features and their thresholds used for splitting are:

1. Destination (categorical feature, used for top-level splits)
2. Month (numerical feature, with various thresholds used for splitting)
 - Thresholds used: 2.5, 5.5, 6.5, 6.2529, 7.5, 10.5