EARTH HAZARD PREDICTION

A report submitted in partial fulfillment of the requirements for the Award of Degree of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

By

YALAMANCHILI NEHASRI

Regd. No.: 20B91A12J1

Under Supervision of Mr. Gundala Nagaraju

Henotic Technology Pvt Ltd, Hyderabad

(Duration: 7th July, 2022 to 6th September, 2022)

DEPARTMENT OF INFORMATION TECHNOLOGY
SAGI RAMA KRISHNAM RAJU ENGINEERING COLLEGE

(An Autonomous Institution)

Approved by AICTE, NEW DELHI and Affiliated to JNTUK, Kakinada

CHINNA AMIRAM, BHIMAVARAM,

ANDHRA PRADESH

SAGI RAMA KRISHNAM RAJU ENGINEERING COLLEGE

(Autonomous)
Chinna Amiram, Bhimavaram

DEPARTMENT OF INFORMATION TECHNOLOGY

CERTIFICATE

This is to certify that the Summer Internship Report “EARTH HAZARD PREDICTION” is the bonafide work done by Mr/Miss YALAMANCHILI NEHASRI bearing Register Number *20B91A1249*at the end of second year second semester at Henotic TechnologyPvt Ltd, Hyderabad from 07.07.2022 to 06.09.2022 inpartial fulfilment of the requirements for the award of the degree of Bachelor of Technology in Information Technology for AIML.

Department Internship Coordinator Dean -T&PCell Head of the Department

Table of Contents

Abstract

In 2016, NASA took on a new responsibility: defending our planet from devastating impacts by asteroids and comets that approach the Earth, or near-Earth objects. That event, which followed the prominent Chelyabinsk meteor explosion in 2013, reflected a growing interest in, and concern about, the threat of celestial impacts. In ancient times, the solar system's small bodies—asteroids and comets—were sometimes seen as ill omens and warnings from the gods. In modern times, they have come to be seen as the solar system's rubble, leftovers from its formation, but were still largely ignored until the late 20th century. Increasingly, they have been seen by scientists as objects worthy of study, by the general public and the U.S. government as potential threats to be mitigated, and by space advocates as future resources. This book tells the fascinating story of these reinterpretations and NASA's role in the society.

1.0Introduction

There is an infinite number of objects in the outer space. Some of them are closer than we think. Even though we might think that a distance of 70,000 Km can not potentially harm us, but at an astronomical scale, this is a very small distance and can disrupt many natural phenomena. These objects/asteroids can thus prove to be harmful. Hence, it is wise to know what is surrounding us and what can harm us amongst those.

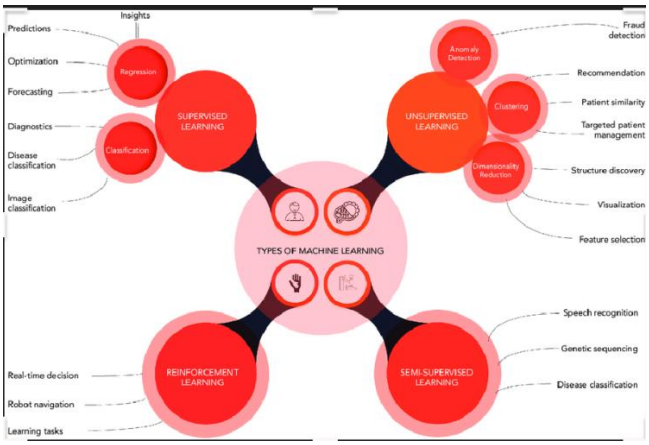
The discovery of Ceres in 1801 provided a new hypothesis for the origins of these rocks, although other, more local explanations retained their currency until about 1860. While conducting observations for what would become a famous star catalog, Italian astronomer Giuseppe Piazzi discovered a new object—Ceres—in an orbit that appeared more planet-like than comet-like. Ceres also fit what is known as the Titius-Bode law. Published in 1766, this rule inferred from the spacing of the inner planets that there should be a planet between Mars and Jupiter; and another beyond Saturn. The 1781 discovery of Uranus approximately where the Titius-Bode law had forecast seemed a confirmation of its accuracy; Ceres, too, was in the right place. Piazzi first thought it might be a comet, but other astronomers doubted that. After the 1802 discovery of Pallas, traveling in a similar orbit, William Herschel declared the two objects to be a new class of celestial body: asteroids.16 The asteroids occupying orbital space between Mars and Jupiter are known collectively today as “main-belt asteroids,” to mark a distinction between them and the many other varieties of asteroid orbits discovered since. They are still the largest known asteroid population, and one, Ceres itself, was later reclassified as a “dwarf planet” by vote of the International Astronomical Union in 2006.17 Main-belt asteroids’ orbits generally keep them far away from Earth, and they are not classed as near-Earth objects. But their orbits can be perturbed, and over the eons some have been shifted into orbits that take them into the inner solar system. The first such near-Earth asteroid to be discovered was 433 Eros, found in 1898 by German astronomer Gustav Witt.

These impact craters are known as “astroblemes.” Bringing the geological community around to the idea that asteroids and comets have in fact left marks on Earth’s surface was the life’s work of Eugene Shoemaker, who also trained the Apollo astronauts in field geology and whose wife, Carolyn, was one of the most prolific discoverers of comets during the era when film was the discovery medium. The recognition that large impacts could influence the evolution of life itself is an extension of what is known as the Alvarez hypothesis, named for a famous 1980 paper by Luis and Walter Alvarez.19 This posits that an impact at the end of the Cretaceous period forced the dinosaurs into extinction. Acceptance of the idea that large impacts have transformed both the surface of Earth and life itself represents a rejection of strict uniformitarianism, or what Stephen J. Gould referred to as “substantive uniformitarianism” in a 1965 essay.20 Three events in close succession brought the potential destructiveness of cosmic impacts to public attention and began moving NEOs into policy salience. The first was the unexpected close flyby of Earth by 1989 FC, discovered after it had already passed by. Discovery of the enormous, buried crater left by the end-Cretaceous impactor in 1990 (or, as we will see in chapter 4, reinterpretation of an already-known structure), was the second. The spectacular collision of Comet Shoemaker-Levy 9 with Jupiter in July 1994, witnessed by ground observatories, by the Hubble Space Telescope, and by the Galileo spacecraft en route to Jupiter, was the third, and most dramatic, event. These led to Congress directing NASA to find at least 90 percent of the 1-kilometer-diameter or larger near-Earth asteroids within 10 years .

1.1. What are the different types of Machine Learning?

There are three types of machine learning. Initially, there were three, but later type added one more type to the ranks of machine learning types.
Thus in total, we have 4 types of machine learning strategies.
These are as under:

1. **Supervised Learning**
2. **Unsupervised Learning**
3. **Semi-supervised Learning**
4. **Reinforcement learning**



1. Supervised Machine Learning

Supervised machine learning is based on supervision. It means in the supervised learning technique, we train the machines using the "labelled" dataset, and based on the training, the machine predicts the output. Here, the labelled data specifies that some of the inputs are already mapped to the output. More preciously, we can say; first, we train the machine with the input and corresponding output, and then we ask the machine to predict the output using the test dataset.

The main goal of the supervised learning technique is to map the input variable(x) with the output variable(y).

Some real-world applications of supervised learning are Risk Assessment, Fraud Detection, Spam filtering, etc.

Categories of Supervised Machine Learning

Supervised machine learning can be classified into two types of problems, which are given below:

- **Classification**
- **Regression**

a) Classification

Classification algorithms are used to solve the classification problems in which the output variable is categorical, such as "**Yes" or No, Male or Female, Red or Blue, etc**. The classification algorithms predict the categories present in the dataset. Some real-world examples of classification algorithms are **Spam Detection, Email filtering, etc.**

Some popular classification algorithms are given below:

- **Random Forest Algorithm**
- **Decision Tree Algorithm**
- **Logistic Regression Algorithm**
- **Support Vector Machine Algorithm**

b) Regression

Regression algorithms are used to solve regression problems in which there is a linear relationship between input and output variables. These are used to predict continuous output variables, such as market trends, weather prediction, etc.

Some popular Regression algorithms are given below:

- **Simple Linear Regression Algorithm**
- **Multivariate Regression Algorithm**
- **Decision Tree Algorithm**
- **Lasso Regression**

2. Unsupervised Machine Learning

Unsupervised learning is different from the Supervised learning technique; as its name suggests, there is no need for supervision. It means, in unsupervised machine learning, the machine is trained using the unlabeleddataset, and the machine predicts the output without any supervision.

In unsupervised learning, the models are trained with the data that is neither classified nor labelled, and the model acts on that data without any supervision.

The main aim of the unsupervised learning algorithm is to group or categories the unsorted dataset according to the similarities, patterns, and differences. Machines are instructed to find the hidden patterns from the input dataset.

Categories of Unsupervised Machine Learning

Unsupervised Learning can be further classified into two types, which are given below:

- **Clustering**
- **Association**

1) Clustering

The clustering technique is used when we want to find the inherent groups from the data. It is a way to group the objects into a cluster such that the objects with the most similarities remain in one group and have fewer or no similarities with the objects of other groups. An example of the clustering algorithm is grouping the customers by their purchasing behaviour.

Some of the popular clustering algorithms are given below:

- **K-Means Clustering algorithm**
- **Mean-shift algorithm**
- **DBSCAN Algorithm**
- **Principal Component Analysis**
- **Independent Component Analysis**

2) Association

Association rule learning is an unsupervised learning technique, which finds interesting relations among variables within a large dataset. The main aim of this learning algorithm is to find the dependency of one data item on another data item and map those variables accordingly so that it can generate maximum profit. This algorithm is mainly applied in Market Basket analysis, Web usage mining, continuous production**,** etc.

Some popular algorithms of Association rule learning are Apriori Algorithm, Eclat, FP-growth algorithm

3. Semi-Supervised Learning

Semi-Supervised learning is a type of Machine Learning algorithm that lies between Supervised and Unsupervised machine learning. It represents the intermediate ground between Supervised (With Labelled training data) and Unsupervised learning (with no labelled training data) algorithms and uses the combination of labelled and unlabeled datasets during the training period.

Although Semi-supervised learning is the middle ground between supervised and unsupervised learning and operates on the data that consists of a few labels, it mostly consists of unlabeled data. As labels are costly, but for corporate purposes, they may have few labels. It is completely different from supervised and unsupervised learning as they are based on the presence & absence of labels.

To overcome the drawbacks of supervised learning and unsupervised learning algorithms, the concept of Semi-supervised learning is introduced. The main aim of semi-supervised learning is to effectively use all the available data, rather than only labelled data like in supervised learning. Initially, similar data is clustered along with an unsupervised learning algorithm, and further, it helps to label the unlabeled data into labelled data. It is because labelled data is a comparatively more expensive acquisition than unlabeled data.

4.Reinforcement Learning

Reinforcement learning works on a feedback-based process, in which an AI agent (A software component) automatically explore its surrounding by hitting & trail, taking action, learning from experiences, and improving its performance**.** Agent gets rewarded for each good action and get punished for each bad action; hence the goal of reinforcement learning agent is to maximize the rewards.

In reinforcement learning, there is no labelled data like supervised learning, and agents learn from their experiences only.

The reinforcement learning process is similar to a human being; for example, a child learns various things by experiences in his day-to-day life. An example of reinforcement learning is to play a game, where the Game is the environment, moves of an agent at each step define states, and the goal of the agent is to get a high score. Agent receives feedback in terms of punishment and rewards.

Due to its way of working, reinforcement learning is employed in different fields such **as** Game theory, Operation Research, Information theory, multi-agent systems.

Categories of Reinforcement Learning

Reinforcement learning is categorized mainly into two types of methods/algorithms:

- **Positive Reinforcement Learning:** Positive reinforcement learning specifies increasing the tendency that the required behaviour would occur again by adding something. It enhances the strength of the behaviour of the agent and positively impacts it.
- **Negative Reinforcement Learning:** Negative reinforcement learning works exactly opposite to the positive RL. It increases the tendency that the specific behaviour would occur again by avoiding the negative condition

1.2.Benefits of Using Machine Learning in NASA nearest earth object

Using artificial intelligence and machine learning in Big Data collections like NASA Earth observation data can eliminate human involvement by quickly and efficiently analyzing years' worth of data. using artificial intelligence technology, a group of researchers calibrate NASA images of the Sun to make them more accurate, improving the solar research data scientists utilize. Since 2010, NASA's Solar Dynamics Observatory, or SDO, has provided high-definition pictures of the Sun. In space exploration, AI has led to the discovery of unmarked galaxies, stars, black holes, as well as the advent of automated flight systems, artificial intelligence with autonomy in navigation, monitoring and management. Using a machine learning model created by ISRO, the imagery can be checked and deforestation detected and reported on more frequently. Among other benefits of this technology are the rapid processing of satellite imagery and reduced time between reports from a year and a month. There are still many objects we need to know about in the deep space so machine learning and deep learning will be used to assist scientists and astronauts to more correctly classify and locate these objects. In consequence, these machines will become better at finding and identifying new objects.

To compete, planet startups rely heavily on the services of data centers. Artificial intelligence has already figured out what black holes and radio galaxies really are. With the aid of AI, we can analyze data from the Gaia space telescope to determine whether new stars shed light on the origin of our galaxy. Satellite images can be interpreted, analyzed, and comprehend by AI. It will help us to identify millions of images produced by space assets with AI. Images taken with a satellite can be analyzed by AI. Images can, in fact, be identified as problematic by this feature. As part of its space research, NASA is constantly developing AI-based applications that automate image analysis for stars, space, and galaxy, developing autonomous space probes in a way where space junk cannot cause injury to astronauts, and making communication networks more secure.

1.3.About Industry (NASA-Nearest Earth Object)

The new agency was to have a distinctly civilian orientation, encouraging peaceful applications in space science. Since its establishment, most US space exploration efforts have been led by NASA, including the Apollo Moon landing missions, the Skylab space station, and later the Space Shuttle.

The National Aeronautics and Space Administration is an independent agency of the US federal government responsible for the civil space program, aeronautics research, and space research.

NASA was established in 1958, succeeding the National Advisory Committee for Aeronautics (NACA), to give the US space development effort a distinctly civilian orientation, emphasizing peaceful applications in space science. Since its establishment, most American space exploration efforts have been led by NASA, including the Apollo Moon landing missions, the Skylab space station, and later the Space Shuttle. NASA is supporting the International Space Station and is overseeing the development of the Orion spacecraft, the Space Launch System, Commercial Crew vehicles, and the planned Lunar Gateway space station. The agency is also responsible for the Launch Services Program, which provides oversight of launch operations and countdown management for uncrewed NASA launches.

NASA's science is focused on better understanding Earth through the Earth Observing System; advancing heliophysics through the efforts of the Science Mission Directorate's Heliophysics Research Program; exploring bodies throughout the Solar System with advanced robotic spacecraft such as New Horizons and researching astrophysics topics, such as the Big Bang, through the Great Observatories and associated programs

1.3.1AI / ML Role in NASA-Nearest Earth Object

Machine Learning is a sub-set of artificial intelligence where computer algorithms are used to autonomously learn from data. Machine learning (ML) is getting more and more attention and is becoming increasingly popular in many other industries. Within the insurance industry, there is more application of ML regarding the claims

2.0 NASA NEAREST EARTH OBJECTS

There is infinite number of objects in the outer space. Some of them are closer than we think. Even though we might think that a distance of 70,000 Km can not potentially harm us, but at an astronomical scale, this is a very small distance and can disrupt many natural phenomena. These objects/asteroids can thus prove to be harmful. Hence, it is wise to know what is surrounding us and what can harm us amongst those. Thus, this dataset compiles the list of NASA certified asteroids that are classified as the nearest earth object

2.1 Main Drivers for AI on NASA-Nearest Earth Objects

Predictive modelling allows for simultaneous consideration of many variablesand quantification of their overall effect. When a large number of claims areanalysed, patterns regarding the characteristics of the claims that drive lossdevelopment begin to emerge.

The following are the main drivers which influencing the Claims Analytics:

- **Policy Characteristics**
  - ✓ Exposures
  - ✓ Limits and Deductibles
  - ✓ Coverages and Perils
- **Insured Characteristics**
  - ✓ Credit Information
  - ✓ Prior loss experience
  - ✓ Payment history
- **Geography based on insured locations**
  - ✓ Auto Repair Costs
  - ✓ Jurisdictional Orientation
  - ✓ Demographics
  - ✓ Crime
- **Agency Characteristics**
  - ✓ Exclusive Agents
  - ✓ Independent Agents

- **Claim information**
  - ✓ FNOL
  - ✓ Claimant data (Credit info, geography, social data, etc.)
  - ✓ Other participants (insured, doctors, lawyers, witnesses, etc.)
  - ✓ Cause, type of Injury/Damage
  - ✓ Injury or damaged object
  - ✓ Coverage
  - ✓ Loss Location
  - ✓ Date and time of Loss and Report
  - ✓ Weather at time & location of loss
- **Details from Prior Claims**
  - ✓ from same insured
  - ✓ from same claimant
  - ✓ from same location
- **Household Characteristics**

2.2Internship Project –Data link

The internship project data has taken from Kaggle and the link is**Link**-https://www.kaggle.com/datasets/sameepvani/nasa-nearest-earth-objects

3.0 AI / ML Modelling and Results

3.1.Your Problem of Statement

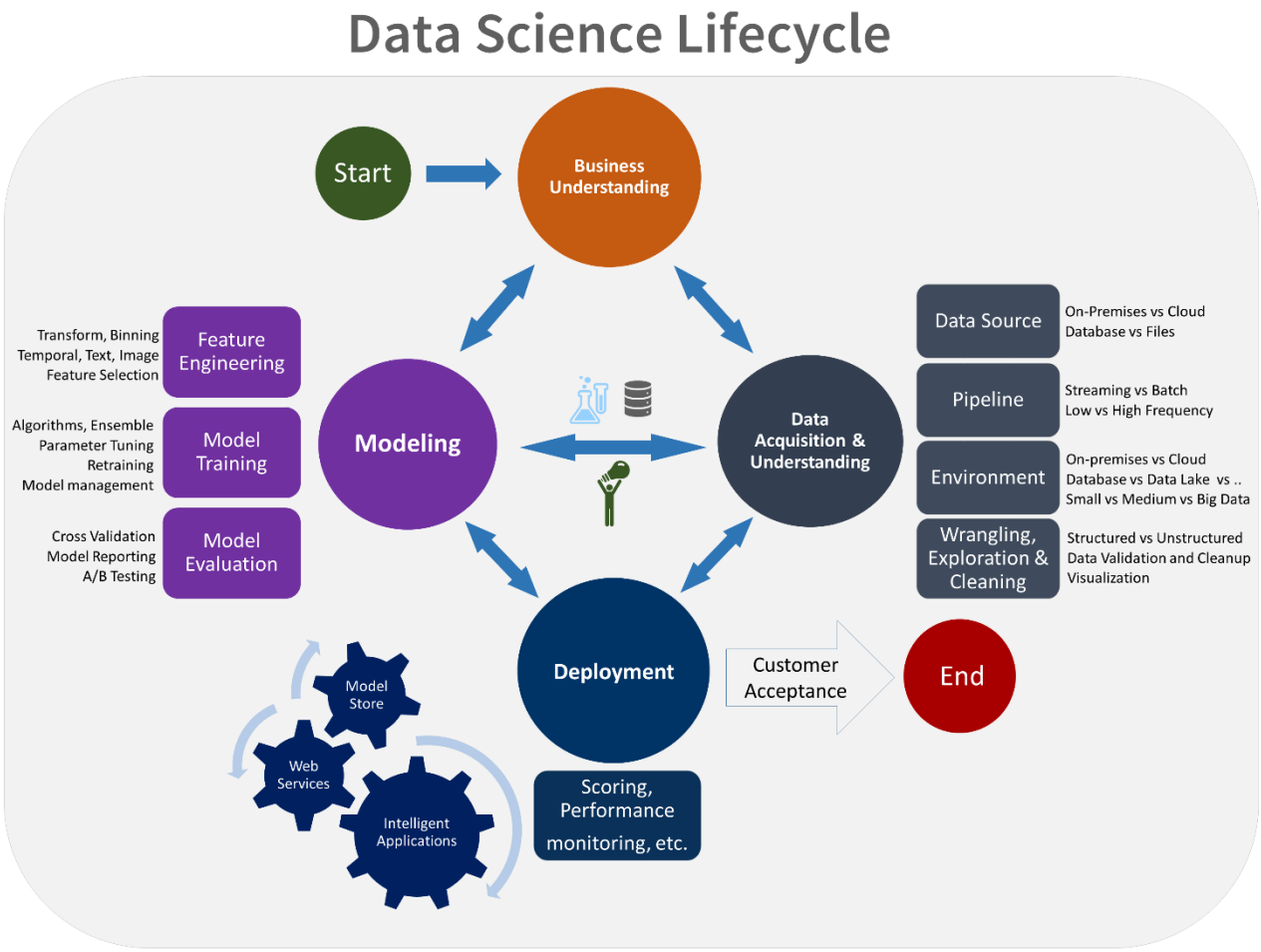
Predictive models are most effective when they are constructed using acompany's own historical claims data since this allows the model to recognize the specific nature of a company's exposure as well as its claims practices.The construction of the model also involves input from the companythroughout the process, as well as consideration of industry leading claimspractices and benchmarks.

Predictive modelling can be used to quantify the impact to the claimsdepartment resulting from the failure to meet or exceed claim service leadingpractice. It can also be used to identify the root cause of claim leakage.Proper use of predictive modelling will allow for potential savings across twodimensions:

- Early identification of claims with the potential for high leakage, thereby allowing for the proactive management of the claim
- Recognition of practices that are unnecessarily increasing claims settlement payments

3.2.Data Science Project Life Cycle

Data Science is a multidisciplinary field of study that combines programming skills, domain expertise and knowledge of statistics and mathematics to extract useful insights and knowledge from data



3.1.1Data Exploratory Analysis

Exploratory data analysis has been done on the data to look for relationship and correlation between different variables and to understand how they impact or target variable.

The exploratory analysis is done for Auto Quote / Policy Conversionwith different parameters and all the charts are presented in **Appendices 6.2 - List of charts (6.2.1 to 6.2.9)**

3.1.2Data Pre-processing

We removed variables which does not affect our target variable(Claimed_Target)as they mayadd noise and also increase our computation time,we checked the data for anomalous data points and outliers.We did principal component analysis on the data set to filter out unnecessary variables and to select only the important variables which have greater correlation with our target variable.

3.2.2.1Check the Duplicate and low variation data

Rows that have identical data are probably useless, if not dangerously misleading during model evaluation.

Here, a duplicate row is a row where each value in each column for that row appears in identically the same order (same column values) in another row.

Rows of duplicate data should probably be deleted from your dataset prior to modeling.

If your dataset simply has duplicate rows, there is no need to worry about preserving the data; it is already a part of the finished dataset and you can merely remove or drop these rows from your cleaned data.

Another approach to the problem of removing columns with few unique values is to consider the variance of the column.

Recall that the variance is a statistic calculated on a variable as the average squared difference ovalues on the sample from the mean.

The variance can be used as a filter for identifying columns to remove from the dataset. A column that has a single value has a variance of 0.0, and a column that has very few unique values will have a small variance value.

3.2.2.2.Identify and address the missing variables

Missing data is defined as the values or data that is not stored (or not present) for some variable/s in the given dataset. In Pandas, usually, missing values are represented by NaN.

Checking the missing values:

The first step in handling missing values is to look at the data carefully and find out all the missing values.dataframe.isnull().sum() will tell about missing values in the entire column.

Figure Out How to Handle the Missing Data:

Analyse each column with missing values carefully to understand the reasons behind the missing values as it is crucial to find out the strategy for handling the missing values. There are 2 primary ways of handling missing values:

1.Deleting the Missing values

2. Imputing the Missing Value

3.2.2.3Handling of Outliers

One of the most important steps as part of data pre-processing is detecting and treating the outliers as they can negatively affect the statistical analysis and the training process of a machine learning algorithm resulting in lower accuracy.

an Outlier is an observation in a given dataset that lies far from the rest of the observations. That means an outlier is vastly larger or smaller than the remaining values in the set.

If our dataset is small, we can detect the outlier by just looking at the dataset. But what if we have a huge dataset, how do we identify the outliers then? We need to use visualization and mathematical techniques

Below are some of the techniques of detecting outliers

- **Boxplots**
- **Z-score**
- **Inter Quantile Range(IQR)**

3.2.2.4 Categorical data and Encoding Techniques

The performance of a machine learning model not only depends on the model and the hyperparameters but also on how we process and feed different types of variables to the model. Since most machine learning models only accept numerical variables, pre-processing the categorical variables becomes a necessary step. We need to convert these categorical variables to numbers such that the model is able to understand and extract valuable information.

Different Encoding Techniques

- **Label Encoding or Ordinal Encoding**
- **One hot Encoding**
- **Dummy Encoding**
- **Effect Encoding**
- **Binary Encoding**
- **BaseN Encoding**
- **Hash Encoding**
- **Target Encoding**

3.2.2.5Feature Scaling

In Data Processing, we try to change the data in such a way that the model can process it without any problems. And Feature Scaling is one such process in which we transform the data into a better version. Feature Scaling is done to normalize the features in the dataset into a finite range.

Why Feature Scaling?

Real Life Datasets have many features with a wide range of values like for example let's consider the house price prediction dataset. It will have many features like no. of. bedrooms, square feet area of the house, etc.

As you can guess, the no. of bedrooms will vary between 1 and 5, but the square feet area will range from 500-2000. This is a huge difference in the range of both features.

Many machine learning algorithms that are using Euclidean distance as a metric to calculate the similarities will fail to give a reasonable recognition to the smaller feature, in this case, the number of bedrooms, which in the real case can turn out to be an actually important metric.

E.g.: Linear Regression, Logistic Regression, KNN

There are several ways to do feature scaling. The below scaling methods are the top 5 of the most commonly used feature scaling techniques.

1. **Absolute Maximum Scaling**
2. **Min-Max Scaling**
3. **Normalization**
4. **Standardization**

5. **Robust Scaling**

3.2.3Selection of Dependent and Independent variables

The dependent or target variable here is 'Dissatisfied' which tells us whether a particular passenger is satisfied or not with the facilities provided by the air ways. The target variable is selected based on our business problem and what we are trying to predict.

The independent variables are selected after doing exploratory data analysis and we used Boruta to select which variables are most affecting our target variable.

3.2.4.Data Sampling Methods

The data we have is highly unbalanced data so we used some sampling methods which are used to balance the target variable so we our model will be developed with good accuracy and precision. We used three Sampling methods

3.2.4.1Stratified sampling

Stratified sampling randomly selects data points from majority class so they will be equal to the data points in the minority class. So, after the sampling both the class will have same no of observations.

It can be performed using strata function from the library sampling.

3.2.4.2Simple random sampling

Simple random sampling is a sampling technique where a set percentage of the data is selected randomly. It is generally done to reduce bias in the dataset which can occur if data is selected manually without randomizing the dataset.

We used this method to split the dataset into train dataset which contains 70% of the total data and test dataset with the remaining 30% of the data.

3.2.5Models Used for Development

We built our predictive models by using the following ten algorithms

3.2.5.1 Model 01 (Logistic Regression)

Logistic uses logit link function to convert the likelihood values to probabilities so we can get a good estimate on the probability of a particular observation to be positive class or negative class. This also gives us p-value of the variables which tells us about significance of each independent variable.

3.2.5.2 Model 02 (Decision Tree Classifier)

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches.

The decisions or the test are performed on the basis of features of the given dataset.

It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.

It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure.

In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm.

A decision tree simply asks a question and based on the answer (Yes/No), it further split the tree into subtrees.

3.2.5.3Model 03 (Random Forest Classifier)

Random forest is an algorithm that consists of many decision trees. It was first developed by Leo Bierman and Adele Cutler. The idea behind it is to build several trees, to have the instance classified by each tree, and to give a "vote" at each class. The model uses a "bagging" approach and the random selection of features to build a collection of decision trees with controlled variance. The instance's class is to the class with the highest number of votes, the class that occurs the most within the leaf in which the instance is placed.

3.2.5.3 Model 04 (Extra Tree Classifier)

This class implements a meta estimator that fits a number of randomized decision trees (a.k.a. extra-trees) on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting.

3.2.5.4 Model 05 (KNN Classifier)

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems

3.2.5.5 Model 06 (Gaussian Naive Bayes)

Naive Bayes is a probabilistic machine learning algorithm used for many classification functions and is based on the Bayes theorem. Gaussian Naive Bayes is the extension of naive Bayes. While other functions are used to estimate data distribution, Gaussian or

normal distribution is the simplest to implement as you will need to calculate the mean and standard deviation for the training data.

What is the Naive Bayes Algorithm?

Naive Bayes is a probabilistic machine learning algorithm that can be used in several classification tasks. Typical applications of Naive Bayes are classification of documents, filtering spam, prediction and so on. This algorithm is based on the discoveries of Thomas Bayes and hence its name.

The name “Naive” is used because the algorithm incorporates features in its model that are independent of each other. Any modifications in the value of one feature do not directly impact the value of any other feature of the algorithm. The main advantage of the Naïve Bayes algorithm is that it is a simple yet powerful algorithm.

3.2.5.6 Model 07 (XGB Classifier)

XG Boost is an implementation of Gradient Boosted decision trees. This library was written in C++. It is a type of Software library that was designed basically to improve speed and model performance. It has recently been dominating in applied machine learning. XG Boost models majorly dominate in many Kaggle Competitions. In this algorithm, decision trees are created in sequential form. Weights play an important role in XG Boost. Weights are assigned to all the independent variables which are then fed into the decision tree which predicts results. The weight of variables predicted wrong by the tree is increased and the variables are then fed to the second decision tree. These individual classifiers/predictors then ensemble to give a strong and more precise model. It can work on regression, classification, ranking, and user-defined prediction problems.

3.2.5.7 Model 08 (Light GBM Classifier)

Light GBM is a gradient boosting framework based on decision trees to increases the efficiency of the model and reduces memory usage.

It uses two novel techniques: Gradient-based One Side Sampling and Exclusive Feature Bundling (EFB) which fulfils the limitations of histogram-based algorithm that is primarily used in all GBDT (Gradient Boosting Decision Tree) frameworks. The two techniques of GOSS and EFB described below form the characteristics of Light GBM Algorithm. They comprise together to make the model work efficiently and provide it a cutting edge over other GBDT frameworks.

Gradient-based One Side Sampling Technique for Light GBM: Different data instances have varied roles in the computation of information gain. The instances with larger gradients (i.e., under-trained instances) will contribute more to the information gain. GOSS keeps those instances with large gradients (e.g., larger than a predefined threshold, or among the top percentiles), and only randomly drop those instances with small gradients to retain the accuracy of information gain estimation. This treatment can lead to a more accurate gain estimation than uniformly random sampling, with the same target sampling rate, especially when the value of information gain has a large range.

Exclusive Feature Bundling Technique for LightGBM:

High-dimensional data are usually very sparse which provides us a possibility of designing a nearly lossless approach to reduce the number of features. Specifically, in a sparse feature space, many features are mutually exclusive, i.e., they never take nonzero values simultaneously. The exclusive features can be safely bundled into a single feature (called an Exclusive Feature Bundle).

3.2.5.8 Model 09 (SVM)

Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.

The goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane.

SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

3.2.5.9 Model 10 (Gradient Boosting Classifier)

This algorithm builds an additive model in a forward stage-wise fashion; it allows for the optimizationof arbitrary differentiable loss functions. In each stage n_classes_ regression trees are fit on the negative gradient of the loss function, e.g., binary or multiclass log loss. Binary classification is a special case where only a single regression tree is induced.

3.3 AI / ML ModelsAnalysis and Final Results

We used our train dataset to build the above models and used our test data to check the accuracy and performance of our models.

We used confusion matrix to check accuracy, Precision, Recall and F1 score of our models and compare and select the best model for given auto dataset of size ~ 272252 policies.

3.3.1Different Model codes

- The Python code for all models sampling technique as follows:

```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.svm import SVC
from sklearn.ensemble import BaggingClassifier
```

```
from sklearn.ensemble import GradientBoostingClassifier
import lightgbm as lgb
# Create objects of classification algorithm with default hyper-parameters
ModelLR = LogisticRegression()
ModelDC = DecisionTreeClassifier()
ModelRF = RandomForestClassifier()
ModelET = ExtraTreesClassifier()
ModelKNN = KNeighborsClassifier(n_neighbors=5)
ModelBAG = BaggingClassifier(base_estimator=None, n_estimators=100, max_samples=1.0, max_features=1.0,
                             bootstrap=True, bootstrap_features=False, oob_score=False, warm_start=False,
                             n_jobs=None, random_state=None, verbose=0)
ModelGB = GradientBoostingClassifier(loss='deviance', learning_rate=0.1, n_estimators=100, subsample=1.0,
                                     criterion='friedman_mse', min_samples_split=2, min_samples_leaf=1,
                                     min_weight_fraction_leaf=0.0, max_depth=3, min_impurity_decrease=0.0,
                                      init=None, random_state=None,
                                     max_features=None, verbose=0, max_leaf_nodes=None, warm_start=False,
                                     validation_fraction=0.1, n_iter_no_change=None, tol=0.0001, ccp_alpha=0.0)
ModelLGB = lgb.LGBMClassifier()
ModelGNB = GaussianNB()
# Evalution matrix for all the algorithms
MM = [ModelLR, ModelDC, ModelRF, ModelET, ModelKNN, ModelBAG, ModelGB, ModelLGB, ModelGNB]
for models in MM:
    # Fit the modEL
    models.fit(X_train, y_train)
    # Predict
    y_pred = models.predict(X_test)
    y_pred_prob = models.predict_proba(X_test)
```

```
    # Print the model na
    print('Model Name: ', models)
    # confusion matrix in sklearn
    from sklearn.metrics import confusion_matrix
    from sklearn.metrics import classification_report
    # actual values
    actual = y_test
    # predicted values
    predicted = y_pred
    # confusion matrix
    matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
    print('Confusion matrix : \n', matrix)
    # outcome values order in sklearn
    tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
    print('Outcome values : \n', tp, fn, fp, tn)
    # classification report for precision, recall f1-score and accuracy
    C_Report = classification_report(actual,predicted,labels=[1,0])
    print('Classification report : \n', C_Report)
    # calculating the metrics
    sensitivity = round(tp/(tp+fn), 3);
    specificity = round(tn/(tn+fp), 3);
    accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
    balanced_accuracy = round((sensitivity+specificity)/2, 3);
    precision = round(tp/(tp+fp), 3);
    f1Score = round((2*tp/(2*tp + fp + fn)), 3);
    # Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
    # A model with a score of +1 is a perfect model and -1 is a poor model
    from math import sqrt
    mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
    MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
    print('Accuracy :', round(accuracy*100, 2),'%')
    print('Precision :', round(precision*100, 2),'%')
    print('Recall :', round(sensitivity*100,2), '%')
```

```
    print('F1 Score :', f1Score)
    print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
    print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
    print('MCC :', MCC)
    # Area under ROC curve
    from sklearn.metrics import roc_curve, roc_auc_score
    print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
    # ROC
    from sklearn.metrics import roc_auc_score
    from sklearn.metrics import roc_curve
    logit_roc_auc = roc_auc_score(actual, predicted)
    fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(X_test)[:,1])
    plt.figure()
    # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
    plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
    plt.plot([0, 1], [0, 1],'r--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")
    plt.savefig('Log_ROC')
    plt.show()
    new_row = {'Model Name' : models,
               'True_Positive' : tp,
               'False_Negative' : fn,
               'False_Positive' : fp,
               'True_Negative' : tn,
               'Accuracy' : accuracy,
               'Precision' : precision,
               'Recall' : sensitivity,
  'F1 Score' : f1Score,
```

```
                'Specificity' : specificity,
                'MCC':MCC,
                'ROC_AUC_Score':roc_auc_score(actual, predicted),
                'Balanced Accuracy':balanced_accuracy}
        CSResults = CSResults.append(new_row, ignore_index=True)
```

3.3.2 Random Forest Python Code

- The Python code for models with simple random forest sampling technique as follows:

```
# To build the 'RandomForestClassifier' model with random sampling
from sklearn.ensemble import RandomForestClassifier
ModelRF1 = RandomForestClassifier(n_estimators=100, criterion='gini', max_depth=None, min_samples_split=2,
                                  min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt',
                                  max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False,
                                  n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None,
                                  ccp_alpha=0.0, max_samples=None)
# Train the model with train data
ModelRF1.fit(X_train,y_train)
# Predict the model with test data set
y_pred = ModelRF1.predict(X_test)
y_pred_prob = ModelRF1.predict_proba(X_test)
# Confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
# confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
```

```
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metri
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy = round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3);
f1Score = round((2*tp/(2*tp + fp + fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
# A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqr
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)
# Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
# ROC Curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(actual, predicted)
fpr, tpr, thresholds = roc_curve(actual, ModelRF1.predict_proba(X_test)[:,1])
plt.figure()
```

```
    # plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
    plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
    plt.plot([0, 1], [0, 1],'r--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")
    plt.savefig('Log_ROC')
    plt.show()
    print('-------------------------------------------------------------------------------------------------')
    #-------------------------------------------------------------------------------------------------
    new_row = {'Model Name' : ModelRF1,
               'True_Positive': tp,
               'False_Negative': fn,
               'False_Positive': fp,
               'True_Negative': tn,
               'Accuracy' : accuracy,
               'Precision' : precision,
               'Recall' : sensitivity,
               'F1 Score' : f1Score,
               'Specificity' : specificity,
               'MCC':MCC,
               'ROC_AUC_Score':roc_auc_score(actual, predicted),
               'Balanced Accuracy':balanced_accuracy}
    RF_HTResults = RF_HTResults.append(new_row, ignore_index=True)
    #-------------------------------------------------------------------------------------------------
```

- The Python code for models with Hyper Parameter Tunning of random forest sampling technique as follows:

```
# To build the 'RandomForestClassifier' model with random sampling with Hyperparametr tuning with RandomizedSearchCV
from sklearn.ensemble import RandomForestClassifier
```

```
ModelRF5 = RandomForestClassifier(n_estimators=2000, criterion='gini', max_depth=110, min_samples_split=2,
                                  min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt',
                                  max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=False, oob_score=False,
                                  n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None,
                                  ccp_alpha=0.0, max_samples=None)
# Train the model with train data
ModelRF5.fit(X_train,y_train)
# Predict the model with test data set
y_pred = ModelRF5.predict(X_test)
y_pred_prob = ModelRF5.predict_proba(X_test)
# Confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted values
predicted = y_pred
# confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy = round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3);
```

```
f1Score = round((2*tp/(2*tp + fp + fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
# A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)
# Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(actual, predicted), 3))
# ROC Curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(actual, predicted)
fpr, tpr, thresholds = roc_curve(actual, ModelRF5.predict_proba(X_test)[:,1])
plt.figure()
# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
print('--------------------------------------------------------------------------------------------------')
```

```
#-------------------------------------------------------------------------------------------------------
--
new_row = {'Model Name' : ModelRF5,
        'True_Positive': tp,
        'False_Negative': fn,
        'False_Positive': fp,
        'True_Negative': tn,
        'Accuracy' : accuracy,
        'Precision' : precision,
        'Recall' : sensitivity,
        'F1 Score' : f1Score,
        'Specificity' : specificity,
        'MCC':MCC,
        'ROC_AUC_Score':roc_auc_score(actual, predicted),
        'Balanced Accuracy':balanced_accuracy}
RF_HTResults = RF_HTResults.append(new_row, ignore_index=True)
#-------------------------------------------------------------------------------------------------------
```

3.3.3 Extra Trees Python code

- The Python code for models with simple Extra Trees sampling technique as follows:

```
# To build the 'ExtraTreesClassifier' model with random sampling along with default hyper parameters values
from sklearn.ensemble import ExtraTreesClassifier
ModelET1 = ExtraTreesClassifier(n_estimators=100, criterion='gini', max_depth=None, min_samples_split=2,
                    min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt',
                    max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=False, oob_score=False,
                    n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None,
                    ccp_alpha=0.0, max_samples=None)
# Train the model with train data
ModelET1.fit(X_train,y_train)
# Predict the model with test data set
y_pred = ModelET1.predict(X_test)
y_pred_prob = ModelET1.predict_proba(X_test)
```

```
# Confusion matrix in sklearn
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted value
predicted = y_pred
# confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy = round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3);
f1Score = round((2*tp/(2*tp + fp + fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between -1 to +1.
# A model with a score of +1 is a perfect model and -1 is a poor model
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%' )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
```

```
print('MCC :', MCC)
# Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(y_test, y_pred), 3))
# ROC Curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, y_pred)
fpr, tpr, thresholds = roc_curve(y_test,ModelET1.predict_proba(X_test)[:,1])
plt.figure()
# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
print('---------------------------------------------------------------------------------------------------
')
new_row = {'Model Name' : ModelET1,
        'True_Positive': tp,
        'False_Negative': fn,
        'False_Positive': fp,
        'True_Negative': tn,
        'Accuracy' : accuracy,
        'Precision' : precision,
        'Recall' : sensitivity,
        'F1 Score' : f1Score,
        'Specificity' : specificity,
        'MCC':MCC,
        'ROC_AUC_Score':roc_auc_score(y_test, y_pred),
        'Balanced Accuracy':balanced_accuracy}
```

```
ET_HTResults = ET_HTResults.append(new_row, ignore_index=True)
#--------------------------------------------------------------------------------------------
```

- The Python code for models with Hyper Parameter Tunning of Extra Tress sampling technique as follows:

```
# To build the 'ExtraTreesClassifier' model with random sampling with Hyperparameter tuning with RandomizedSearchCV
from sklearn.ensemble import ExtraTreesClassifier
ModelET5 = ExtraTreesClassifier(n_estimators=600, criterion='entropy', max_depth=120, min_samples_split=2,
                                min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt',
                                max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=False, oob_score=False,
                                n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None,
                                ccp_alpha=0.0, max_samples=None)
# Train the model with train data
ModelET5.fit(X_train,y_train)
# Predict the model with test data set
y_pred = ModelET5.predict(X_test)
y_pred_prob = ModelET5.predict_proba(X_test)
# Confusion matrix in sklear
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
# actual values
actual = y_test
# predicted
predicted = y_pred
# confusion matrix
matrix = confusion_matrix(actual,predicted, labels=[1,0],sample_weight=None, normalize=None)
print('Confusion matrix : \n', matrix)
# outcome values order in sklearn
tp, fn, fp, tn = confusion_matrix(actual,predicted,labels=[1,0]).reshape(-1)
print('Outcome values : \n', tp, fn, fp, tn)
# classification report for precision, recall f1-score and accuracy
C_Report = classification_report(actual,predicted,labels=[1,0])
```

```
print('Classification report : \n', C_Report)
# calculating the metrics
sensitivity = round(tp/(tp+fn), 3);
specificity = round(tn/(tn+fp), 3);
accuracy = round((tp+tn)/(tp+fp+tn+fn), 3);
balanced_accuracy = round((sensitivity+specificity)/2, 3);
precision = round(tp/(tp+fp), 3);
f1Score = round((2*tp/(2*tp + fp + fn)), 3);
# Matthews Correlation Coefficient (MCC). Range of values of MCC lie between
-1 to +1.
# A model with a score of +1 is a perfect model and -1 is a poor mode
from math import sqrt
mx = (tp+fp) * (tp+fn) * (tn+fp) * (tn+fn)
MCC = round(((tp * tn) - (fp * fn)) / sqrt(mx), 3)
print('Accuracy :', round(accuracy*100, 2),'%')
print('Precision :', round(precision*100, 2),'%')
print('Recall :', round(sensitivity*100,2), '%')
print('F1 Score :', f1Score)
print('Specificity or True Negative Rate :', round(specificity*100,2), '%'  )
print('Balanced Accuracy :', round(balanced_accuracy*100, 2),'%')
print('MCC :', MCC)
# Area under ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
print('roc_auc_score:', round(roc_auc_score(y_test, y_pred), 3))
# ROC Curv
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
logit_roc_auc = roc_auc_score(y_test, y_pred)
fpr, tpr, thresholds = roc_curve(y_test,ModelET5.predict_proba(X_test)[:,1])
plt.figure()
# plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot(fpr, tpr, label= 'Classification Model' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
```

```
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
print('-----------------------------------------------------------------------------------------------------')
new_row = {'Model Name' : ModelET5,
        'True_Positive': tp,
        'False_Negative': fn,
        'False_Positive': fp,
        'True_Negative': tn,
        'Accuracy' : accuracy,
        'Precision' : precision,
        'Recall' : sensitivity,
        'F1 Score' : f1Score,
        'Specificity' : specificity,
        'MCC':MCC,
        'ROC_AUC_Score':roc_auc_score(y_test, y_pred),
        'Balanced Accuracy':balanced_accuracy}
ET_HTResults = ET_HTResults.append(new_row, ignore_index=True)
```

Stratified Sampling: Random Forest model performance is good, by considering the confused matrix, highest accuracy (1.0) &good F1 score (1.0).This is because random forest uses bootstrap aggregation which can reduce bias and variance in the data and can leadsto good predictions withclaims dataset.

Simple Random Sampling: Artificial Neural Networks / Random Forest are out performed Logistic Regression model, by considering the confused matrix, highest accuracy (1.0) &good F1 score (1.0).This is because Artificial Neural Networks havehidden and complex patterns between different variables and can leads to good predictions with claims dataset.

4 Conclusions and Future work

The model results in the following order by considering the model accuracy, F1 score and RoC AUC score.

1) **GradientBoostingClassifier** with Stratified and Random Sampling
2) **SVC** with Simple Random Sampling
3) **Extra Tree Classifier** with Simple Random Sampling

We recommend model - GradientBoostingClassifier with Stratified and Random Sampling technique as a best fit for the given dataset. We considered GradientBoostingClassifier because The objective of Gradient Boosting classifiers is **to minimize the loss, or the difference between the actual class value of the training example and the predicted class value**.

	Model Name	True Positive	False Negative	False Positive	True Negative	Accuracy	Precision	Recall	F1 Score	Specificity	MCC	ROC_AUC_Score
0	LogisticRegression()	281	2373	534	24063	0.893	0.345	0.106	0.162	0.978	MCC	0.542084
1	DecisionTreeClassifier()	1169	1485	3702	20895	0.81	0.24	0.44	0.311	0.849	MCC	0.644981
2	(DecisionTreeClassifier(max_features='auto', r...	640	2014	676	23921	0.901	0.486	0.241	0.322	0.973	MCC	0.606831
3	(ExtraTreeClassifier(random_state=1443359620),...	517	2137	443	24154	0.905	0.539	0.195	0.286	0.982	MCC	0.588395
4	KNeighborsClassifier()	738	1916	1017	23580	0.892	0.421	0.278	0.335	0.959	MCC	0.618362
5	GaussianNB()	1890	764	4414	20183	0.81	0.3	0.712	0.422	0.821	MCC	0.76634
6	SVC(probability=True)	215	2439	61	24536	0.908	0.779	0.081	0.147	0.998	MCC	0.539265
7	(DecisionTreeClassifier(random_state=146828511...	719	1935	923	23674	0.895	0.438	0.271	0.335	0.962	MCC	0.616693
8	([DecisionTreeRegressor(criterion='friedman_ms...	525	2129	243	24354	0.913	0.684	0.198	0.307	0.99	MCC	0.593968
9	LGBMClassifier()	844	1810	803	23794	0.904	0.512	0.318	0.392	0.967	MCC	0.642682

Model Name	True Positive	False Negative	False Positive	True Negative	Accuracy	Precision	Recall	F1 Score	Specificity	MCC	ROC_AUC_Score	Balanced Accuracy
LogisticRegression()	281	2373	534	24063	0.893	0.345	0.106	0.162	0.978	MCC	0.542084	0.542
DecisionTreeClassifier()	1169	1485	3702	20895	0.81	0.24	0.44	0.311	0.849	MCC	0.644981	0.644
ssifier(max_features='auto', r...	640	2014	676	23921	0.901	0.486	0.241	0.322	0.973	MCC	0.606831	0.607
random_state=1443359620),...	517	2137	443	24154	0.905	0.539	0.195	0.286	0.982	MCC	0.588395	0.588
KNeighborsClassifier()	738	1916	1017	23580	0.892	0.421	0.278	0.335	0.959	MCC	0.618362	0.618
GaussianNB()	1890	764	4414	20183	0.81	0.3	0.712	0.422	0.821	MCC	0.76634	0.766
SVC(probability=True)	215	2439	61	24536	0.908	0.779	0.081	0.147	0.998	MCC	0.539265	0.54
er(random_state=146828511...	719	1935	923	23674	0.895	0.438	0.271	0.335	0.962	MCC	0.616693	0.616
essor(criterion='friedman_ms...	525	2129	243	24354	0.913	0.684	0.198	0.307	0.99	MCC	0.593968	0.594
LGBMClassifier()	844	1810	803	23794	0.904	0.512	0.318	0.392	0.967	MCC	0.642682	0.642

5 References

https://www.nasa.gov/sites/default/files/atoms/files/a_history_of_near-earth_object_research_tagged.pdf

https://www.nasa.gov/audience/foreducators/postsecondary/features/F_Near_Earth_Program.html#:~:text=Near%2DEarth%20Objects%20(NEOs)%20are%20comets%20and%20asteroids%20that,to%20enter%20the%20Earth's%20neighborhood.

https://www.kaggle.com/datasets/sameepvani/nasa-nearest-earth-objects

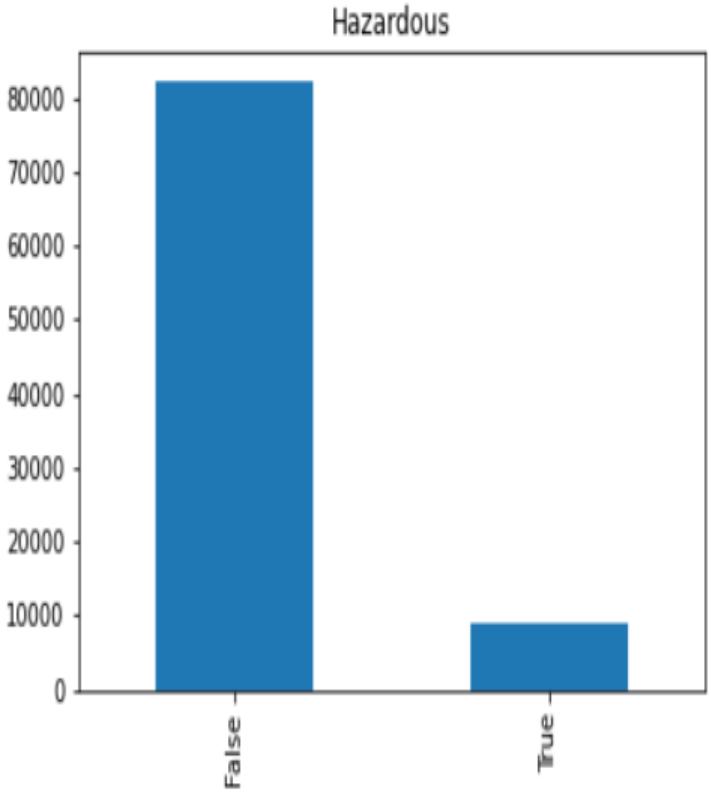
6.0 Appendices

6.1 python code Results

```
Model Name:  GradientBoostingClassifier()
Confusion matrix :
 [[  525  2129]
 [  243 24354]]
Outcome values :
 525 2129 243 24354
Classification report :
              precision    recall  f1-score   support

           1       0.68      0.20      0.31      2654
           0       0.92      0.99      0.95     24597

    accuracy                           0.91     27251
   macro avg       0.80      0.59      0.63     27251
weighted avg       0.90      0.91      0.89     27251

Accuracy : 91.3 %
Precision : 68.4 %
Recall : 19.8 %
```

6.2**Charts**

6.2.1chart 01:about hazardous



6.2.2 chart 02: about all the data