1.0 Introduction

With the increasing power of computer technology, companies and institutions can nowadays store large amounts of data at reduced cost. The amount of available data is increasing exponentially and cheap disk storage makes it easy to store data that previously was thrown away. There is a huge amount of information locked up in databases that is potentially important but has not yet been explored. The growing size and complexity of the databases makes it hard to analyse the data manually, so it is important to have automated systems to support the process. Hence there is the need of computational tools able to treat these large amounts of data and extract valuable information.

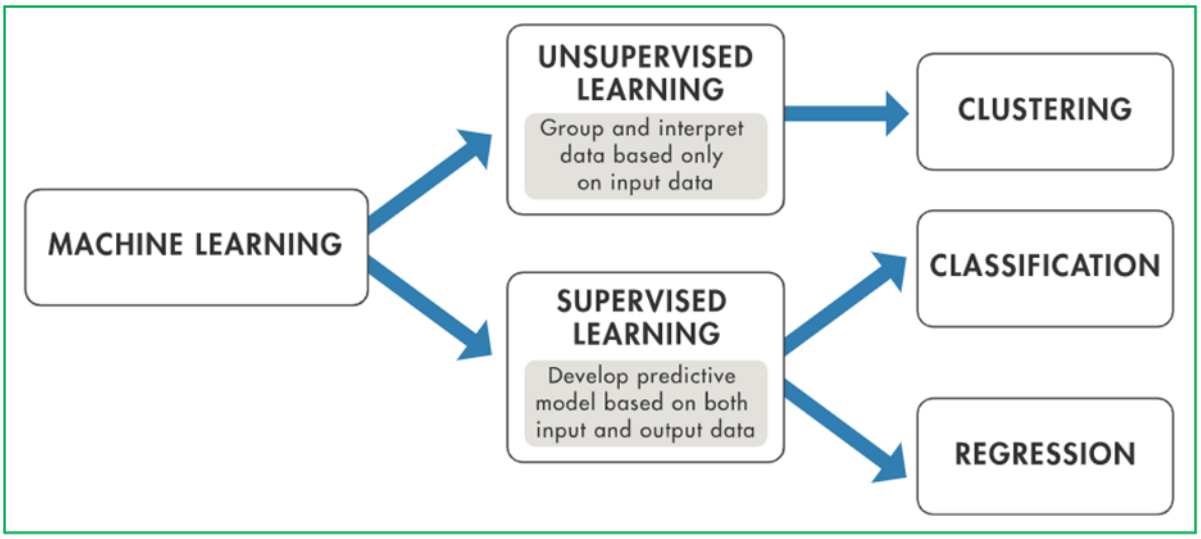
In this context, Data Mining provides automated systems capable of processing large amounts of data that are already present in databases. Data Mining is used to automatically extract important patterns and trends from databases seeking regularities or patterns that can reveal the structure of the data and answer business problems. Data Mining includes learning techniques that fall into the field of Machine learning. The growth of databases in recent years brings data mining at the forefront of new business technologies.

Most of the organizations or companies have a formal performance evaluation system in which employee job performance is graded on a regular basis, usually once or twice a year. Employee performance evaluation is designed to assess everyone’s contribution to the organization. The performance of individuals against organizational goals determines whether the organization meets its goals. The basic objectives of performance evaluations are two-fold: firstly, to reward employees for meeting organizational objectives and secondly to identify which objectives are not met and to develop action plans to ensure they are achieved in the future. It also helps remind workers what their managers expect in the workplace. Moreover, the purpose of any performance review is to be sure employees know how they can develop their talents and energies and use them more effectively to contribute to the organization's success. Also, for the employees to grow and contribute effectively requires a good match between the employee's talents and energies and the demands of the position. A good performance evaluation system can prominently benefit an organization. It helps employee behaviour toward organizational aims by permitting employees know what is expected for them, and it yields information for making employment decisions, such as those regarding pay raises, promotion or releases. Developing and implementing an effective system is not easy task. An Employee can improve their performance by way of monitoring the progression of their performance. Machine learning algorithms i.e. decision tree of data mining technique etc can be used to find out the key characteristics of future prediction of an organization.

To sustain this company’s ongoing business, they want to determine the characteristics of their performing employees and help the underperforming employees to be better. To help the Company, the researchers wants to analyze the performance of their internal operations. It will give some insights how the employees being monitored and evaluated on their work. The objective of the study is to determine the characteristics of the excellent and good performing employees. Also, to create a model that will predict employees who will perform well.

1.1. What are the different types of Machine Learning?

Machine learning uses two types of techniques: **Supervised learning**, which trains a model on known input and output data so that it can predict future outputs, and **Unsupervised learning**, which finds hidden patterns or intrinsic structures in input data.



Supervised Learning

Supervised machine learning builds a model that makes predictions based on evidence in the presence of uncertainty. A supervised learning algorithm takes a known set of input data and known responses to the data (output) and trains a model to generate reasonable predictions for the response to new data. Use supervised learning if you have known data for the output you are trying to predict.

Supervised learning uses **Regression** and **Classification** techniques to develop predictive models.

Regression techniques predict continuous responses - for example, changes in temperature or fluctuations in power demand. Typical applications include electricity load forecasting and algorithmic trading.

Use regression techniques if you are working with a data range or if the nature of your response is a real number, such as temperature or the time until failure for a piece of equipment.

Common regression algorithms include linear model, nonlinear model, stepwise regression, Gradient Descent Regression, Support Vector Regression, Ridge and Lasso Regressions.

Classification techniques predict discrete responses - for example, whether an email is genuine or spam, or whether a tumour is cancerous or benign. Classification models classify input data into categories. Typical applications include medical imaging, speech recognition, and credit scoring.

Use classification if your data can be tagged, categorized, or separated into specific groups or classes. For example, applications for hand-writing recognition use classification to recognize letters and numbers. In image processing and computer vision, unsupervised pattern recognition techniques are used for object detection and image segmentation.

Common algorithms for performing classification include support vector machine (SVM), boosted and bagged decision trees, k-nearest neighbour, Naïve Bayes, discriminant analysis, logistic regression, and neural networks.

Using Supervised Learning to Predict Heart Attacks: Suppose clinicians want to predict whether someone will have a heart attack within a year. They have data on previous patients, including age, weight, height, and blood pressure. They know whether the previous patients had heart attacks within a year. So, the problem is combining the existing data into a model that can predict whether a new person will have a heart attack within a year.

Unsupervised Learning

Unsupervised learning finds hidden patterns or intrinsic structures in data. It is used to draw inferences from datasets consisting of input data without labelled responses.

Clustering is the most common unsupervised learning technique. It is used for exploratory data analysis to find hidden patterns or groupings in data. Applications for cluster analysis include gene sequence analysis and market research.

For example, if a cell phone company wants optimize the locations where they build cell phone towers, they can use machine learning to estimate the number of clusters of people relying on their towers. A phone can only talk to one tower at a time, so the team uses clustering algorithms to design the best placement of cell towers to optimize signal reception for groups, or clusters, of their customers.

Common algorithms for performing clustering include k-means and k-medoids, Apriori algorithms, hierarchical clustering, Gaussian mixture models and hidden Markov models.

1.2. Benefits of Using Machine Learning in Employee promotion prediction:

Employee Promotion means the ascension of an employee to higher ranks. It involves an increase in salary, position, responsibilities, status, and benefits. This aspect of the job drives employees the most—the ultimate reward for dedication and loyalty towards an organization. Employee Promotion plays a big role in Employee Satisfaction. It aids in **employee engagement**, **boosts morale**, **reduces absenteeism**, and ultimately in productivity.

Employee Promotion is also a helpful tool for reducing attrition retention. Attrition has been a dominant problem in all companies. By practicing proper **performance appraisals** and employee promotion, this problem can be tackled.

So, for the reasons stated above, companies must promote deserving employees in timely intervals. Promoting equals progress, and progress is what's best for business. So for promoting deserved employees we need machine learning. By using maching learning techniques we can predict whose who promoted which leads to company profits.

1.3. About Industry or organization :

The organization or industry has been going through massive transformations in the past couple of years. A promotion is a powerful communication tool about what is valued within an organization. Thus, industries promote employees for the growth of organization. a promotion in industries or organizations must be available to employees who play any role in the contribution of work and value. With a new competent organization budding every other week, the market competition is getting more cut-throat than ever.

Examples of a Promotion in industries:

- HR Assistant receives a promotion to HR Generalist
- HR Generalist receives a promotion to a dual role of HR Generalist and Employee Development Coordinator
- HR Generalist is given a promotion to HR Manager
- HR Manager is given a promotion to Manager of Human Resources and Administration
- HR Manager is promoted to HR Director

1.3.1 AI / ML Role in Employee promotion prediction:

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 all industries and organizations, there is more application of ML regarding the promotion.

2.0 Employee promotion prediction data set:

The data was extracted internally from the HRIS Team of the Company. The data given was in Excel format and is composed of structured data, and the definitions have been discussed and validated by the H.R. Manager.

The following details for an employee is given in the dataset :-

- employee_id: Unique ID for the employee
- department: Department of employee
- region: Region of employment (unordered)
- education: Education Level
- gender: Gender of Employee
- recruitment_channel: Channel of recruitment for employee
- no_ of_ trainings: no of other trainings completed in the previous year on soft skills, technical skills, etc.
- age: Age of Employee
- previous_ year_ rating: Employee Rating for the previous year
- KPIs met(1 if >80%)-total KPIs met in the tenure
- length_ of_ service: Length of service in years
- awards_ won: if awards won during the previous year then 1 else 0
- avg_ training_ score: Average score in current training evaluations
- is_promoted: (Target) Recommended for promotion

2.1. Main Drivers for AI Promotion Analysis

Predictive modelling allows for simultaneous consideration of many variables and quantification of their overall effect. When a large number of promotions are analysed, patterns regarding the characteristics of the promotions that drive loss development begin to emerge.

The following are the main drivers which influencing the promotion:

- performance
- age
- service length
- previous year rating
- total score

2.2. Internship Project - Data Link

The internship project data has taken from Kaggle and the link is https://www.kaggle.com/datasets/ifeanyichukwunwobodo/employee-promotion-prediction

3.0 AI / ML Modelling and Results

3.1. Your Problem of Statement

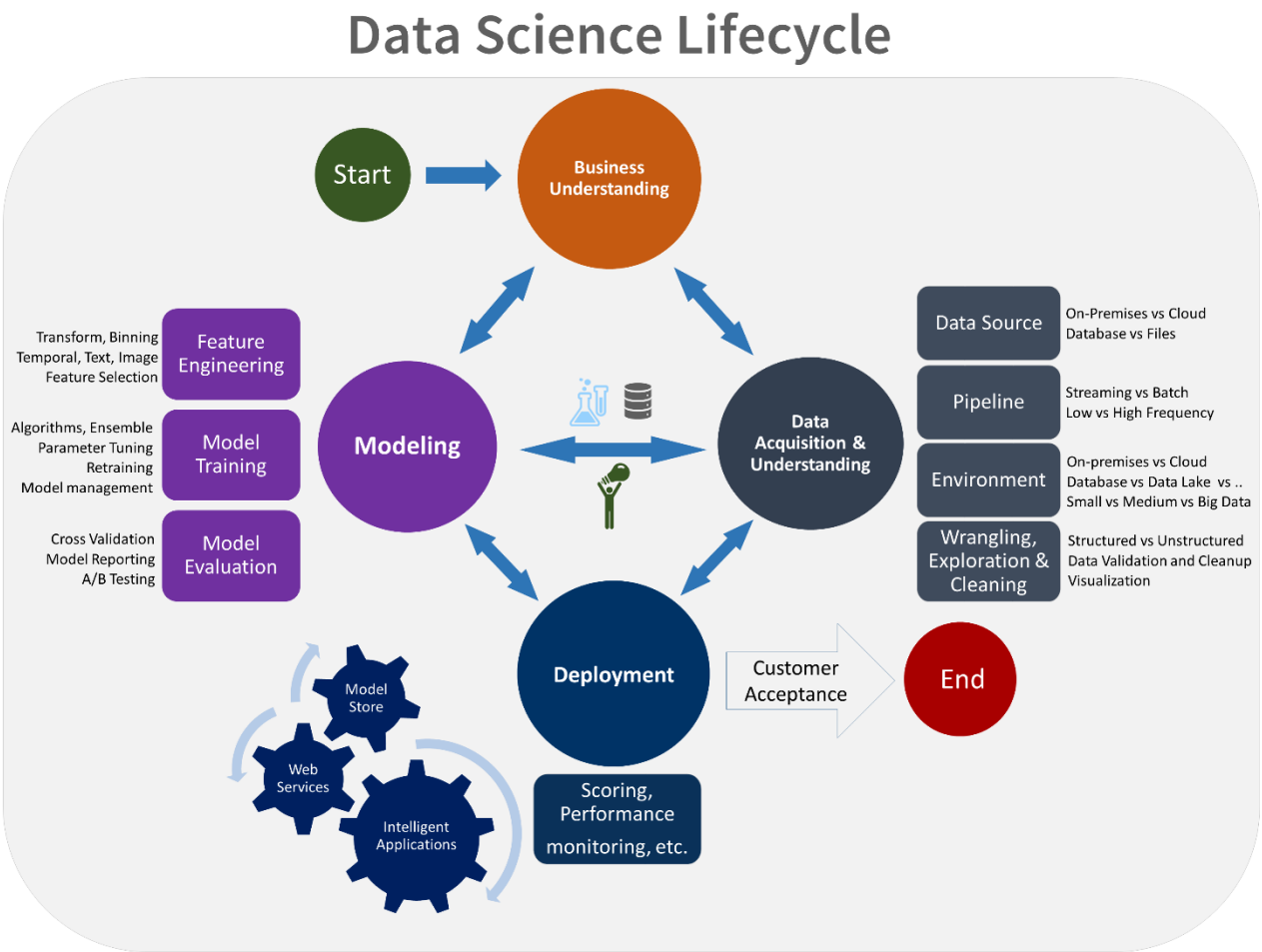
Predictive models are most effective when they are constructed using a company's own historical claims data since this allows the model to recognize the specific nature of a company's exposure as well as its claims practices. The construction of the model also involves input from the company throughout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

Predictive modelling can be used to quantify the impact to the claims department resulting from the failure to meet or exceed claim service leading practices. It can also be used to identify the root cause of claim leakage. Proper use of predictive modelling will allow for potential savings across two dimensions:

- 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.



3.2.1 Data 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.

Variable *Promoted* is the outcome variable, indicating whether an employee was promoted into the SES position after working year 10 (1 = promoted, 0 = not promoted). In order for the analysis to be consistent, we only included employees with GS or GM pay plan. 482 out of 27,932 employees were promoted.

3.2.2 Data Pre-processing

We removed variables which does not affect our target variable(is_promoted)as they may add 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.

3.2.2.1. Check the Duplicate and low variation data

Duplicate Values: When two features have the same set of values

Duplicate Index: When the value of two features are different, but they occur at the same index

Steps to delete duplicates:

1.Use the get duplicate features functions to get all the constant features.

2. Store all the duplicate features as a list for removing from the dataset.

3.Drop all such features from the dataset.

3.2.2.2. Identify and address the missing variables

1) Find some columns with missing values in your dataset.

2) Use the Imputer class so you can impute missing values

3) Add columns with missing values to your predictors.

Reasons of having Missing Values in Data set,

- Unavailability of Data
- Loss of Data
- Data Entry Error
- Incomplete Forms etc.

Treatment of Missing Values is very Important as Machine Learning Predictive Models cannot work with Missing Values.

3.2.2.3. Handling of Outliers

- The presence of outliers in a classification or regression dataset can result in a poor fit and lower predictive modelling performance.
- Outliers Detection and Treatment becomes necessary in some cases.
- Outliers can be Found in Numeric Columns of the Data.
- Columns in Dataset which might have Outliers
    - Average Training Score
    - Length of Service.
- Box plots can be used for identifying Outliers in Data.

3.2.2.4. Categorical data and Encoding Techniques

● Encoding categorical data is a process of converting categorical data into integer format so that the data with converted categorical values can be provided to the different models.

● A categorial data or non-numerical data – where variable has value of observations in form of categories, further it can have two types-

a. Nominal

b. Ordinal

● We already know that Machine Learning algorithms working only with Numeric Data.

● So, we have to encode our Object Data and Convert that into Numeric. so that Machine Learning Model can accept our Data.

● There are five Columns in our Dataset which needs encoding

- ✓ Department
- ✓ Region
- ✓ Education
- ✓ Recruitment_channel
- ✓ Gender

3.2.2.5. Feature Scaling

● Feature scaling is a method used to normalize the range of independent variables or features of data. In data processing, it is also known as data normalization and is generally performed during the data preprocessing step.

● Feature Scaling basically helps to normalise the data within a particular range. Sometimes, It also helps in speeding up the calculations in an algorithm.

3.2.3 Selection of Dependent and Independent variables

The dependent or target variable here is is_promoted Target which tells us a particular policy holder has filed a promoted or not the target variable is selected based on some factors 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

3.2.4.1. Stratified 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.

3.2.4.2. Simple 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.

3.2.5 Models 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.The 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)

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.

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.

3.2.5.3. Model 03(Random Forest Classifier)

Random forest is an algorithm that consists of many decision trees. It was first developed by Leo Breiman 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.

The error of the forest depends on:

- Trees correlation: the higher the correlation, the higher the forest error rate.
- The strength of each tree in the forest. A strong tree is a tree with low error. By using trees that classify the instances with low error the error rate of the forest decreases.

3.2.5.4. Model 04(Extra Tree Classifier)

Extremely Randomized Trees Classifier (Extra Trees Classifier) is a type of ensemble learning technique which aggregates the results of multiple de-correlated decision trees collected in a "forest" to output its classification result. In concept, it is very similar to a Random Forest Classifier and only differs from it in the manner of construction of the decision trees in the forest.

3.2.5.5. Model 05(KNN Classifier)

K-Nearest Neighbour 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. K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data. It is also called a lazy learner algorithm because it does

not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset. KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

3.2.5.6. Model 06(GaussianNB)

Naïve Bayes is a probabilistic machine learning algorithm used for many classification functions and is based on the Bayes theorem. Gaussian Naïve Bayes is the extension of naïve 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.

3.2.5.7. Model 07(SVC)

Support Vector Machine(SVM) is a supervised machine learning algorithm used for both classification and regression. Though we say regression problems as well its best suited for classification. The objective of SVM algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane. It becomes difficult to imagine when the number of features exceeds three.

3.2.5.8. Model 08(XGB Classifier)

XGB 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. XGB models majorly dominate in many Kaggle Competitions. In this algorithm, decision trees are created in sequential form. Weights play an important role in XGB. 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.

3.2.5.9. Model 09(LGBM Classifier)

LGBM 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 LGBM 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 LGBM: 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.

3.2.5.10. Model 10(Bagging Classifier)

A Bagging classifier is an ensemble meta-estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions (either by voting or by averaging) to form a final prediction. Such a meta-estimator can
typically be used as a way to reduce the variance of a black-box estimator (e.g., a decision tree), by introducing randomization into its construction procedure and then making an ensemble out of it. Each base classifier is trained in parallel with a training set which is generated by randomly drawing, with replacement, N examples(or data) from the original training dataset – where N is the size of the original training set. Training set for each of the base classifiers is independent of each other. Many of the original data may be repeated in the resulting training set while others may be left out.

Bagging reduces overfitting (variance) by averaging or voting, however, this leads to an increase in bias, which is compensated by the reduction in variance though.

3.2.5.11. Model 11(Gradient Boosting Classifier)

Gradient boosting classifier is a set of machine learning algorithms that include several weaker models to combine them into a strong big one with highly predictive output. Models of a kind are popular due to their ability to classify datasets effectively.

Using **gradient boost for classification** we discover the initial prediction for every patient in the **log (odds)**.

3.3. AI / ML Models Analysis 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.

3.3.1 Different Model codes

```
# Build the Calssification models and compare the results

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)

ModelSVM = SVC(probability=True)

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()

MM2=[ModelLR,ModelDC,ModelRF,ModelET,ModelKNN,ModelSVM,modelBAG,ModelGB,ModelLGB, ModelGNB]
```

```
for models in MM2:
 # Train the model training dataset
 models.fit(x_train, y_train)
 # Prediction the model with test dataset
y_pred = models.predict(x_test)
 y_pred_prob = models.predict_proba(x_test)
# Print the model name
 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 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, y_pred), 3))
# ROC Curve
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve
Model_roc_auc = roc_auc_score(actual, y_pred)
fpr, tpr, thresholds = roc_curve(actual, models.predict_proba(x_test)[:,1])
plt.figure()
plt.plot(fpr, tpr, label= 'Classification Model' % Model_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' : 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, y_pred),
                'Balanced Accuracy':balanced_accuracy}
CSResults= CSResults.append(new_row, ignore_index=True)
#---------------------------------------------------------------------
```

3.3.2 LGBM Python code

- The Python code for models with stratified sampling technique as follows:

```
 # Training the lightgbm model on the Training set
 import lightgbm as lgb
# Build the model
modelLGB = lgb.LGBMClassifier()
# Fit the model with train data
modelLGB.fit(x_train,y_train)
# Predict the model with test data set
y_pred = modelLGB.predict(x_test)
y_pred_prob = modelLGB.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(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,modelLGB.predict_proba(x_test)[:,1])
plt.figure()
# plt.plot
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('----------------------------------------------------------------------------------------------------')
```

3.3.3 Decision tree classifier

```
# To build the 'Decision Tree' model with random sampling
from sklearn.tree import DecisionTreeClassifier
CustChurnDT = DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini', max_depth=None, max_features=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None,min_samples_leaf=1,
min_samples_split=2,min_weight_fraction_leaf=0.0,random_state=None, spliter='best')
# Train the model with train data
CustChurnDT.fit(x_train,y_train)
# Predict the model with test data set
y_pred = CustChurnDT.predict(x_test)
y_pred_prob = CustChurnDT.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(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, CustChurnDT.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.show()
print('-----------------------------------------------------------------------------------------------------')
-----------------------------------------------------------------------------------------------------------------
# To get feature importance
from matplotlib import pyplot
importance = CustChurnDT.feature_importances_
# Summarize feature importance
for i,v in enumerate(importance):
 print('Feature: %0d, Score: %.5f' % (i,v))
# Plot feature importance
pyplot.bar([x for x in range(len(importance))], importance)
pyplot.show()
```

Stratified Sampling: LGBM Classifier model performance is good, by considering the confused matrix, highest accuracy (0.946) & good F1 score (0.513).

4.0 Conclusions and Future work

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

1) **LGBM Classifier** with Stratified and Random Sampling
2) **Decision tree classifier** with Stratified and Random Sampling
3) **Decision tree Regressor** with Simple Random Sampling

We recommend model – **LGBM Classifier** with Stratified Sampling technique as a best fit for the give n BI claims dataset. We considered LGBM Classifier because it uses bootstrap aggregation which can reduce bias and variance in the data and can leads to good predictions with employee promotion prediction datase

In [46]:
```
CSResults
```

Out[46]:

	Model Name	True_Positive	False_Negative	False_Positive	True_Negative	Accuracy	Precision	Recall	F1 Score	Specificity	M
0	LogisticRegression()	92	1241	54	15056	0.921	0.63	0.069	0.124	0.996	(
1	DecisionTreeClassifier()	586	747	979	14131	0.895	0.374	0.44	0.404	0.935	0.
2	(DecisionTreeClassifier(max_features='sqrt', r...	341	992	39	15071	0.937	0.897	0.256	0.398	0.997	(
3	(ExtraTreeClassifier(random_state=128917307), ...	316	1017	65	15045	0.934	0.829	0.237	0.369	0.996	0.
4	KNeighborsClassifier()	182	1151	207	14903	0.917	0.468	0.137	0.211	0.986	0.
5	SVC(probability=True)	98	1235	20	15090	0.924	0.831	0.074	0.135	0.999	0.
6	(DecisionTreeClassifier(random_state=844253368...	453	880	67	15043	0.942	0.871	0.34	0.489	0.996	0.
7	([DecisionTreeRegressor(criterion='friedman_ms...	376	957	18	15092	0.941	0.954	0.282	0.435	0.999	0.
8	LGBMClassifier()	471	862	31	15079	0.946	0.938	0.353	0.513	0.998	0.
9	GaussianNB()	161	1172	216	14894	0.916	0.427	0.121	0.188	0.986	0.

5.0 References

- The dataset has taken from Kaggle and the link

  https://www.kaggle.com/datasets/ifeanyichukwunwobodo/employee-promotion-prediction

- Information gathered from

  https://www.simplilearn.com/10-algorithms-machine-learning-engineers-need-to-know-article

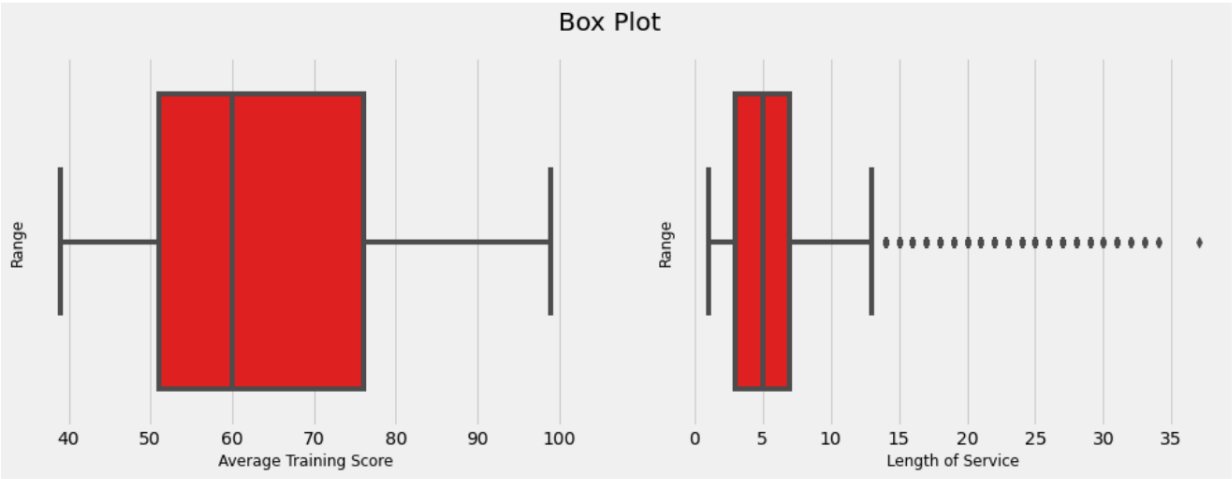
  https://cdn.dataisgood.com/wp-content/uploads/2020/10/12225729/Employee-Promotion-Prediction.pdf

6.0 Appendices

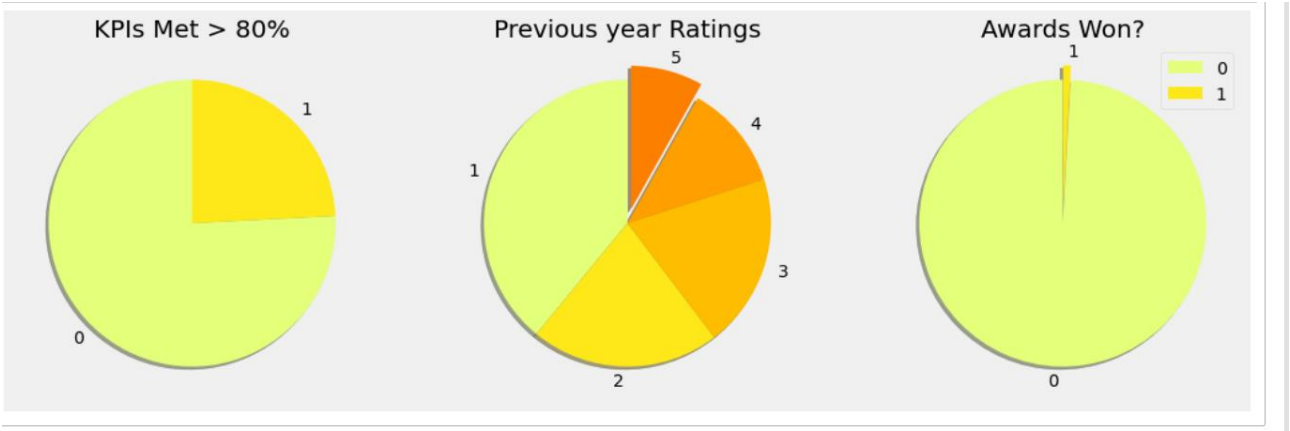
6.1 List of Charts

6.1.1 Chart 01: to check the Target Class Balance

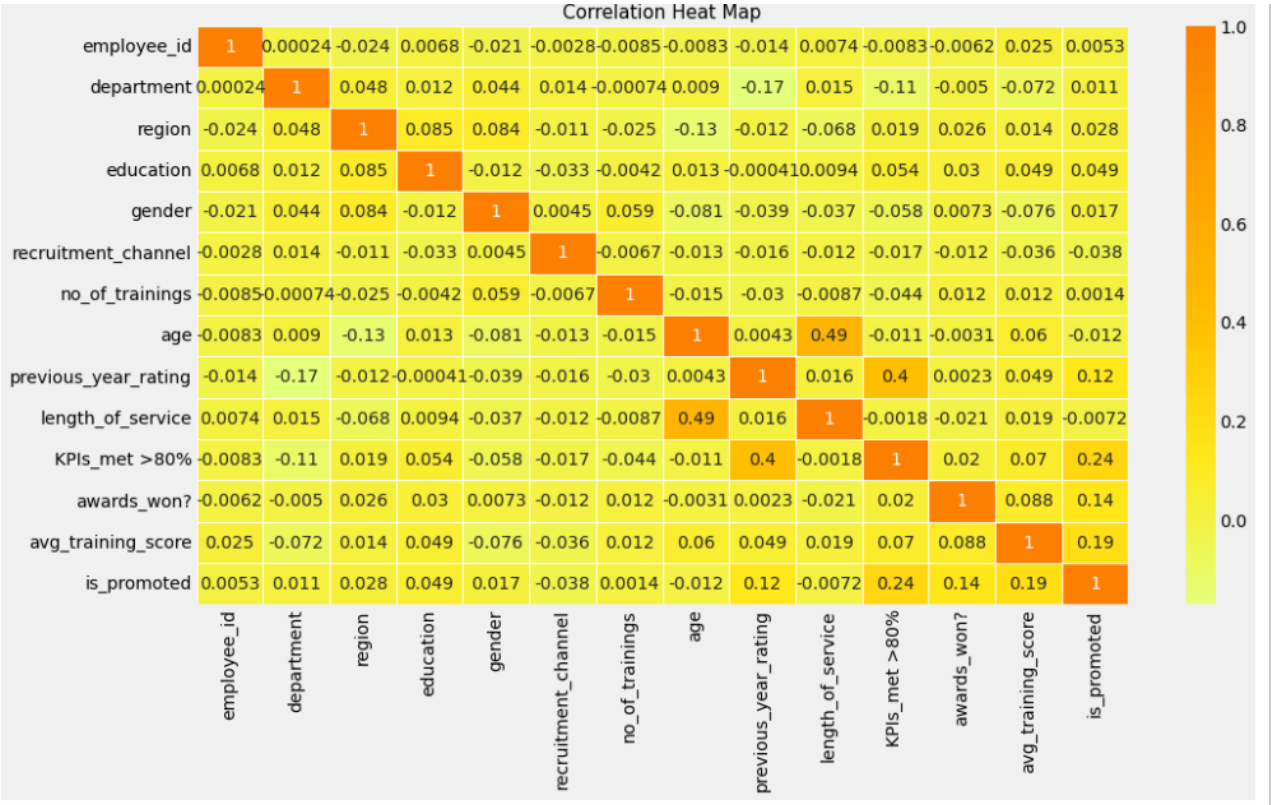
6.1.2 Chart 02: to check the boxplots for the columns where we suspect for outliers



6.1.3 Chart 03: to plot pie chart for the columns where we have very few categories



6.1.4 Chart 04:to check the Heat Map for the Data with respect to correlation.



6.1.5 Chart 04:to check the relation of Departments and Promotion when they won awards