“Employee Attrition Prediction using Artificial Neural Networks”

Abstract

My study focuses on the benefits of using an **Artificial Neural Network (ANN)** model to predict **Employee attrition**. Employee attrition is a big concern for businesses today, and anticipating employee turnover can assist businesses in identifying at-risk personnel and taking proactive actions to retain them. In this research, an ANN model was trained using a dataset comprising various employee parameters such as age, education, job satisfaction, Monthly Income, and so on to predict the likelihood of an employee quitting the organization.

The dataset was preprocessed to manage variables, and categorical variables were converted to numerical values by one-hot encoding. The values are also scaled between 0 and 1 using Sklearn's MinMaxScaler. The ANN model was trained using the Keras API using the Tensorflow library. The model was trained on the preprocessed dataset using binary cross entropy as the loss function and Adam optimizer.

To avoid the model being biased towards the dominant class in the sample, the dataset was balanced using **SMOTE (Synthetic Minority Over-sampling TEchnique)** .

The model's performance was assessed using multiple measures like accuracy, precision, recall, and F1 score. On the test set, the model had an accuracy of 87%, indicating its effectiveness in predicting employee attrition. To understand the benefit of using an ANN model for prediction, the classification report of the ANN model is compared to the classification reports of other models trained on the same dataset, such as Logistic Regression, Naive Bayes, KNN (K Nearest Neighbors) and SVM (Support Vector Machines).

Along with classification reports, the ROC-AUC (Area under the curve of Receiver Operating Characteristic) of the models are compared to evaluate the difference in performances.

The comparison shows that the ANN model has higher scores in terms of classification report and ROC-AUC.
Organizations can utilize the project's findings to identify at-risk individuals and take the required steps to retain them, lowering employee turnover and boosting organizational performance.

INTRODUCTION

Employee attrition is the pace at which employees depart a company. High staff attrition can be a big burden for organizations, resulting in a variety of issues. First and foremost, employee churn can be costly due to increasing recruitment and training expenditures. Furthermore, employee attrition can lead to decreased productivity because new employees need time to learn their roles and responsibilities.

Furthermore, high levels of employee attrition can result in knowledge and expertise loss, especially if the departing employees are experienced and skilled. This might have an adverse effect on work quality and client service. Additionally, high employee attrition rates can lead to low employee morale, as remaining employees may become demotivated as a result of increased workload and a perceived lack of job security.

As a result, organizations must understand the variables that lead to employee attrition and take proactive steps to prevent it. An ANN model, for example, can assist organizations in identifying at-risk individuals and taking the required actions to retain them, lowering employee turnover and enhancing organizational performance.

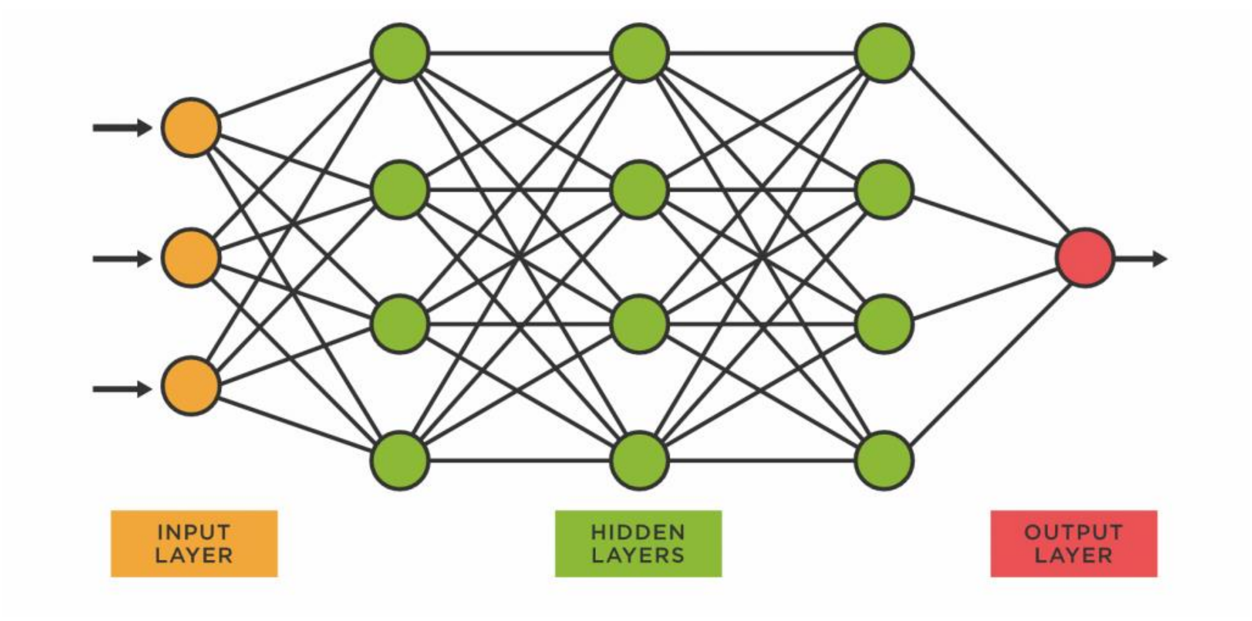
An artificial neural network (ANN) is a class of machine learning algorithms modeled after the structure and function of the human brain. It consists of layers of interconnected nodes or neurons, each of which performs simple computations on its inputs.
ANNs are used for various applications such as classification, regression, and pattern recognition. They are particularly well suited for tasks that process large amounts of data, such as image and speech recognition.

The ANN training process adjusts the connection weights between neurons according to the input data in order to minimize a cost function that measures the difference between the network's output and the desired output. Once trained, the ANN can be used to make predictions on new data.

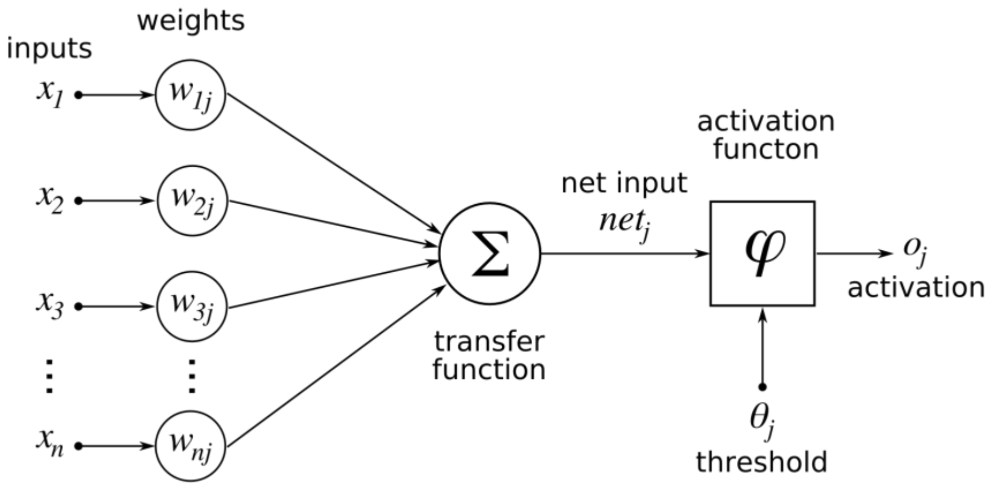
ANNs are widely used in areas such as computer vision, speech recognition, and natural language processing. They are also used in various industrial and commercial applications such as predicting equipment failures in manufacturing and optimizing supply chain logistics.

The structure of ANNs is usually divided into three main components:

An input layer, one or more hidden layers, and an output layer. The input layer consists of neurons that receive input data, and the output layer makes the network's final predictions. Hidden layers between the input and output layers perform intermediate computations on the input data.



The core component of ANN is an artificial neuron. Each neuron receives inputs from multiple other neurons, multiplies them by their assigned weights, sums them, and passes the sum to one or more neurons. Some artificial neurons may apply an activation function to the output before passing it to the next variable.



The ANN training process adjusts the weights of the connections between neurons according to the input data in order to minimize a cost function that measures the difference between the network's output and the desired output. This is usually done using an algorithm such as backpropagation. Backpropagation propagates the error from the output layer through the network, adjusting the weights as it progresses.
Once trained, an ANN can be used to make predictions on new data by passing the data through the input layer and computing the output of the output layer.

Activation functions

An artificial neural network (ANN) activation function is a mathematical formula that determines the output of a neural network. They are used to introduce non-linearities in the output of neural networks, giving them the ability to learn complex patterns in the data. An activation function takes an input signal and applies a mathematical function to produce an output signal. There are several activation functions commonly used in ANNs, including :

Sigmoid function:
The sigmoid function is a general activation function that maps arbitrary input values to values between 0 and 1. Used in binary classification problems.

ReLU function (rectified linear units):
ReLU is another common activation function commonly used in ANNs. Maps negative input values to 0 and leaves positive input values unchanged.

Tanh function (hyperbolic tangent):
The tanh function maps each input value to a value between -1 and 1. Often used in binary classification problems.

Softmax function:
Softmax functions are often used in the output layer of ANNs that perform multiclass classification. Map output values to a probability distribution over multiple classes. Choosing an appropriate activation function is important for ANN design. Choosing an activation function depends on the nature of the problem, the neural network architecture, and the type of data used.

Artificial Neural Networks (ANN) have several advantages over traditional machine learning algorithms. Some of the advantages are:

- **Non-linearity**: ANNs are capable of learning non-linear relationships between inputs and outputs, which traditional machine learning algorithms may struggle with.

- **Adaptability:** ANNs can adapt to changes in input patterns, which makes them useful for applications such as time-series forecasting.

- **Fault tolerance:** ANNs are robust to noisy data and can tolerate missing data.

- **Generalization:** ANNs can generalize well to new, unseen data, which makes them useful for real-world applications.

- **Parallel processing:** ANNs can be implemented on parallel processing architectures, which allows for faster training and prediction.

- **Feature extraction:** ANNs can automatically extract relevant features from raw input data, which can reduce the need for manual feature engineering.

Overall, the advantages of using ANNs make them a powerful tool for solving complex problems in a variety of fields, including computer vision, natural language processing, and predictive analytics.

Objective of the study

To study the effectiveness of Artificial Neural Networks in predicting Employee Attrition of an organization.
To study the significance of choosing Artificial Neural Networks over other traditional models for prediction
To understand the importance of balancing the datasets before fitting into the model

Scope of the study

The scope of study is to understand the functioning of Artificial Neural Networks. Also, to understand the advantages and challenges in training an ANN model. This study is conducted to understand whether opting ANN as a prediction model over other traditional ML algorithms will bring improvement to the accuracy, precision, recall and f1 score or not.

Limitations of the study

The dataset used is a fictitious dataset created by IBM data scientists, hence this model is not convenient to predict Employee Attrition in a real scenario.
Realistic data set should be used instead of fictitious dataset for better predictions.

LITERATURE REVIEW

The article "**Prediction of heart disease using ANNs**" by **A. Khademi et al. (2019)** proposes the use of artificial neural networks (ANNs) to predict heart disease. The authors train and test the ANN model using patient information data sets, including demographic, clinical, and laboratory variables. They compare the performance of ANN models to logistic regression and decision tree models. In this study, we found

that the ANN model outperforms other models in terms of accuracy, sensitivity, specificity, and area under the receiver operating characteristic curve (AUC-ROC). The authors suggest that the ANN model can be used as a tool to predict heart disease in a clinical setting, potentially contributing to early diagnosis and treatment.
Overall, this study demonstrates the potential utility of his ANN model in medical diagnostic and predictive tasks, especially when traditional statistical models may not perform well.

A study by **Y. Wang et al**, "**ANN-based Stock Prediction System**" **(2019)** propose an artificial neural network (ANN)-based system for stock price prediction. The authors state that stock market forecasting is a difficult task due to the high volatility and complexity of the stock market. They argue that ANNs are suitable tools for this task due to their ability to learn from past data and capture non-linear relationships between variables. The study uses a dataset of historical stock prices and various technical indicators as inputs to the ANN. The authors experiment with different ANN architectures and hyperparameters and compare the system's performance with other machine learning algorithms.
The results show that the ANN-based system outperforms other machine learning algorithms in terms of prediction accuracy. The authors conclude that ANNs are promising tools for stock price prediction, and further research is needed to improve prediction accuracy by integrating additional features and investigating more advanced neural network architectures. It suggests that you can concentrate.

"**ANN-Based Traffic Prediction for Smart Cities**" by **H. Lin et al. (2021)** propose a neural network-based approach for predicting traffic flows in smart cities. The authors argue that traditional statistical models cannot accurately capture the complex dynamics of urban traffic, so neural networks can provide a more accurate and flexible approach to traffic prediction.
This study uses historical traffic data, weather information, and vacation schedules as inputs to a neural network model. The authors experiment with different neural network architectures and hyperparameters to optimize prediction accuracy. They report that their approach outperforms traditional statistical models and achieves high accuracy in predicting traffic flows in large urban networks.
This study highlights the potential of neural networks in smart city applications and demonstrates the feasibility of using historical data to predict traffic patterns. The

authors suggest that their approach can be extended to other smart city areas such as energy management, public safety, and healthcare.

The Article "**Predicting COVID-19 Using Artificial Neural Networks**" by **Hassanien et al**. **(2021)** discusses the use of artificial neural networks (ANNs) to predict COVID-19 infection rates. This study uses data from various sources, including the World Health Organization, to build and train a neural network model that can predict the number of his COVID-19 cases in various regions.
This study highlights the effectiveness of ANNs in predicting COVID-19 infection rates, with the model achieving high accuracy in predicting the number of cases. The authors discuss the potential application of his ANN models in predicting the spread of his COVID-19 in various regions and the potential for these models to support the development of public health policies and strategies. also explains. Overall, this study highlights the potential of ANNs in predicting COVID-19 infection rates and their utility in informing public health policies and strategies. It also highlights the importance of leveraging data from multiple sources to create accurate and effective models.

DATA DESCRIPTION

The **IBM HR Analytics Employee Attrition and Performance** dataset is downloaded from **kaggle.com**.
This is a fictitious dataset created by IBM data scientists. It contains information about an employee of a fictional company, including factors such as age, education, job duties, salary, his work-life balance, performance ratings, and job satisfaction. This dataset was created with the goal of predicting employee turnover, a common challenge faced by many organizations.

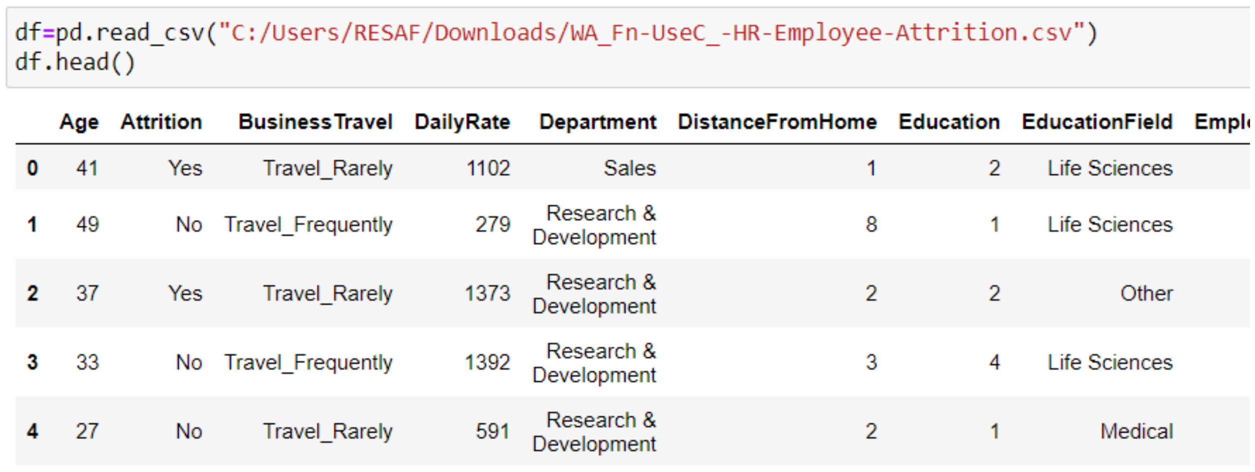
The dataset consists of 35 columns and 1,470 rows of data. Each row corresponds to a unique employee, and columns provide information about various factors that can affect employee attrition. The dataset is labeled and each employee's turnover status (yes or no) is provided as a target variable.

#	Column	DType	Description
1	Age	int64	Employee age in years
2	Attrition	int64	Whether the employee has left the company (Yes) or is still employed (No)
3	BusinessTravel	object	How often the employee travels for business (rarely, frequently, or not at all)
4	DailyRate	int64	Daily rate of pay for the employee
5	Department	object	Department the employee works in (sales, research and development, or human resources)
6	DistanceFromHome	int64	Distance from the employee's home to their workplace
7	Education	int64	Employee's level of education (1 = Below College, 2 = College, 3 = Bachelor, 4 = Master, 5 = Doctor)
8	EducationField	object	Field of study for the employee's highest level of education (life sciences, medical, marketing, technical degree, human resources, other)
9	EmployeeCount	int64	always 1 for all observations
10	EmployeeNumber	int64	unique ID for each employee

11	EnvironmentalSatisfaction	int64	employee's satisfaction with their work environment (1 = low, 2 = medium, 3 = high, 4 = very high)
12	Gender	object	employee's gender (Male or Female)
13	HourlyRate	int64	hourly rate of pay for the employee
14	JobInvolvement	int64	employee's level of involvement in their job (1 = low, 2 = medium, 3 = high, 4 = very high)
15	JobLevel	int64	employee's job level (1 = entry level, 2 = mid-level, 3 = senior level, 4 = manager level, 5 = director level)
16	JobRole	object	employee's job role (sales executive, research scientist, laboratory technician, manufacturing director, healthcare representative, manager, sales representative, research director, human resources)
17	JobSatisfaction	int64	employee's satisfaction with their job (1 = low, 2 = medium, 3 = high, 4 = very high)
18	MaritalStatus	object	employee's marital status (single, married, divorced)
19	MonthlyIncome	int64	employee's monthly income in USD

20	MonthlyRate	int64	monthly rate of pay for the employee
21	NumCompaniesWorked	int64	number of companies the employee has worked for in the past
22	Over18	object	whether the employee is over 18 years old (Y or N)
23	OverTime	object	whether the employee works overtime (Yes or No)
24	PercentSalaryHike	int64	percentage increase in the employee's salary compared to the previous year
25	PerformanceRating	int64	employee's performance rating (1 = low, 2 = good, 3 = excellent, 4 = outstanding)
26	RelationshipSatisfaction	int64	employee's satisfaction with their relationships at work (1 = low, 2 = medium, 3 = high, 4 = very high)
27	StandardHours	int64	always 80 for all observations
28	StockOptionLevel	int64	level of stock options the employee has (0 = none, 1 = low, 2 = medium, 3 = high)
29	TotalWorkingYears	int64	total number of years the employee has worked
30	TrainingTimeLastYear	int64	number of times the employee was trained in the past year
31	WorkLifeBalance	int64	employee's perceived

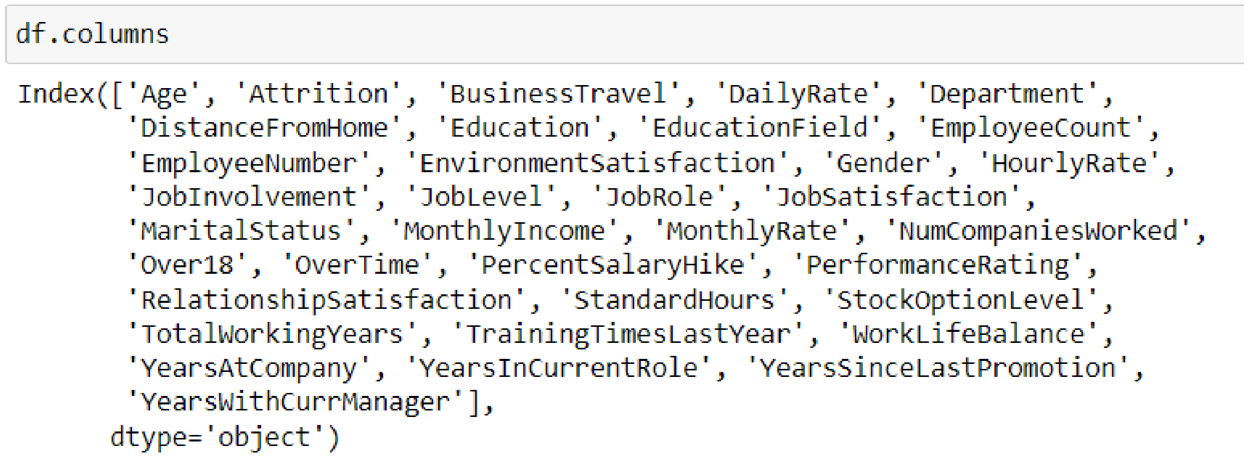
			balance between work and personal life (1 = low, 2 = medium, 3 = high, 4 = very high)
32	YearsAtCompany	int64	number of years the employee has worked at the company
33	YearsInCurrentRole	int64	number of years the employee has been in their current role
34	YearsSinceLastPromotion	int64	number of years since the employee was last promoted
35	YearsWithCurrManager	int64	number of years the employee has worked under their current



```
df=pd.read_csv("C:/Users/RESAF/Downloads/WA_Fn-UseC_-HR-Employee-Attrition.csv")
df.head()
```

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	Empl
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	

Loading and viewing the dataset in Python



```
df.columns
```

```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField', 'EmployeeCount',
       'EmployeeNumber', 'EnvironmentSatisfaction', 'Gender', 'HourlyRate',
       'JobInvolvement', 'JobLevel', 'JobRole', 'JobSatisfaction',
       'MaritalStatus', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'Over18', 'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```

Columns in the Dataset

CONCEPTUAL FRAMEWORK

Artificial Neural Network:

Artificial Neural Networks (ANNs) are machine learning techniques inspired by the structure and function of biological neural networks. It consists of multiple interconnected processing nodes that work together to learn and recognize complex patterns in data.

The operation of an ANN involves several steps starting with an input layer that receives raw data. Input data is then transformed and processed through a series of hidden layers where a neural network applies weights and biases to the input and computes predictions. The output layer produces the final neural network output used for prediction, classification, or other tasks. During the learning process, the neural network adjusts weights and biases based on the error between predicted and actual outputs. This process, called backpropagation, allows the neural network to learn from mistakes and improve accuracy over time.

ANNs have a wide range of applications, including image and speech recognition, natural language processing, and predictive analytics. It has proven to be an effective tool for solving complex problems and making accurate predictions, especially in areas where traditional machine learning algorithms may fail.

Logistic Regression:

Logistic regression is a statistical technique for analyzing the relationship between a binary dependent variable and one or more independent variables. It is commonly used in predictive modeling of classification problems. The goal of logistic regression is to find the best model that describes the relationship between a binary dependent variable and an independent variable by estimating the probability that the dependent variable takes on a particular value.

A logistic regression model uses the logistic function (also called the sigmoid function) to transform the output of a linear regression model into probability values. Probability values are between 0 and 1 and can be interpreted as the probability of a particular outcome. If the probability is greater than or equal to 0.5, the outcome is predicted in the positive class, otherwise it is predicted in the negative class.
Logistic regression estimates the coefficients of the independent variables that maximize the probability of observing the data. Coefficients are estimated using maximum likelihood estimation. This estimation finds a set of coefficients that make the predicted probabilities as close as possible to the observed probabilities. A logistic regression model is then used to predict the probability that the dependent variable will take a particular value for new data based on the values of the independent variables.

K-Nearest Neighbors (KNN):

K-Nearest Neighbors (KNN) is a supervised machine learning algorithm that can be used for both classification and regression tasks. It is a non-parametric algorithm, meaning that it does not make any assumptions about the underlying distribution of the data.

The working of KNN can be summarized in the following steps:

1. Load the data: Load the dataset into the program and preprocess it as required.

2. Define the value of k: Choose a value for k, which represents the number of nearest neighbors to be considered for classification or regression.

3. Calculate the distance: Calculate the distance between the new data point and each of the training data points.

4. Sort the distances: Sort the distances in ascending order to identify the k-nearest neighbors.

5. Calculate the output: For classification, the output is calculated by assigning the new data point to the class that is most common among its k-nearest neighbors. For regression, the output is calculated as the average of the values of its k-nearest neighbors.

6. Evaluate the model: Evaluate the performance of the model using various metrics such as accuracy, precision, recall, and F1-score.

KNN is a simple yet powerful algorithm that can be used for a wide range of applications, including image recognition, recommendation systems, and anomaly detection. However, it can be computationally expensive for large datasets and may not perform well with high-dimensional data. Therefore, it is important to choose the value of k and preprocess the data appropriately to achieve optimal results.

Naive Bayes:

Naive Bayes is a probabilistic classification algorithm based on Bayes' theorem, which states that the probability of a hypothesis (such as a class label) given some observed evidence (such as features) is proportional to the probability of the evidence given the hypothesis multiplied by the prior probability of the hypothesis.

In the context of classification, Naive Bayes assumes that the features are conditionally independent given the class label, which means that the presence or absence of one feature does not affect the presence or absence of another feature given the class label. This assumption simplifies the computation of the probability of the evidence given the hypothesis, and allows Naive Bayes to scale well to high-dimensional feature spaces.

The working method of Naive Bayes involves building a probability model for each class label based on the training data, which involves estimating the prior probability of the class label and the conditional probability of each feature given the class label. Once the model is built, the classification of a new instance involves computing the posterior probability of each class label given the observed features, and selecting the class label with the highest posterior probability as the predicted label.

Naive Bayes is commonly used for text classification tasks, such as spam detection and sentiment analysis, but it can also be applied to other types of data with discrete or continuous features. It is a simple and fast algorithm that can perform well even with small amounts of training data, but its accuracy may suffer if the independence assumption is violated or if there is strong correlation among the features.

Support Vector Machine (SVM):

Support vector machines (SVMs) are powerful supervised machine learning algorithms that can be used for both classification and regression tasks. The goal of SVM is to find an optimal decision boundary (hyperplane) that can separate different classes in a data set.

The way SVM works is to find a hyperplane that maximizes the distance between the closest points of different classes. These closest points are called support vectors, hence the name support vector machine. Margin is the distance between the decision line and the closest point for each class. SVM can handle both linearly separable and nonlinearly separable datasets. For nonlinearly separable datasets, SVM uses a technique called kernel tricks to map the input data into a high-dimensional space. This allows SVM to find hyperplanes that can separate the classes in the transformed space.

The main advantage of SVM is that it can handle high-dimensional datasets with relatively few training samples. Additionally, SVMs can handle noisy data and datasets with outliers and have a solid theoretical foundation. However, SVMs are sensitive to the choice of kernel function and its parameters, and can be computationally intensive on large datasets.

METHODOLOGY

The proposed work analyzes the respective dataset to detect the most influential features that affect the prediction and builds a predictive model according to the following phases.

1. Gathering employees' data: IBM HR Analytics Employee Attrition and Performance dataset has been used.

2. Preprocessing the collected data: Data is prepared to be utilized by the predictive model. The data needs to undergo several processes in order to be ready to fit to the model.

3. Analyzing the dataset: The most important features that push an employee to leave the organization are detected. Correlations are plotted and studied.

4. Balancing the dataset: Since the dataset is not already balanced, it is necessary to be equalized. An unbalanced dataset is not ideal to fit into a model, because it might create biases in predictions. Hence it is treated with a technique called SMOTE to make it balanced.

5. Building the predictive model: The suitable configuration for the model is identified and selected to increase the accuracy, precision, recall and f1 score off the model.

6. Evaluation of the model: The model is evaluated using the classification metrics such as precision, recall, f1 score and accuracy. The AUC-ROC Score is also used to evaluate the performance of the model. 80%:20% train-test set used for system evaluation.

Libraries utilized

Pandas:

Pandas is an open-source data manipulation library for the Python programming language. It provides data structures and functions for efficiently handling large datasets and performing various data analysis tasks.
The two primary data structures provided by Pandas are Series and DataFrame. A Series is a one-dimensional array-like object that can hold any data type, whereas a DataFrame is a two-dimensional table-like data structure with columns of potentially different data types.

Pandas allows for easy data loading and cleaning, merging and joining data, reshaping data, and handling missing data. It also has functions for data filtering, aggregation, and transformation, making it a powerful tool for data analysis and manipulation. Overall, Pandas is widely used by data scientists and analysts for its efficient and user-friendly data handling capabilities.

NumPy:

NumPy is a Python library that stands for 'Numerical Python'. It is a fundamental package for scientific computing in Python and provides a powerful N-dimensional array object, which can be used for performing various mathematical and logical operations. NumPy has functions for linear algebra, random number generation, Fourier transform, and much more. It also provides interfaces to libraries for numerical integration, optimization, signal processing, and machine learning. NumPy arrays are more efficient than Python lists for numerical operations and can be easily manipulated using built-in functions and operations. NumPy is a widely used library in the data science and scientific computing community.

Matplotlib:

Matplotlib is a plotting library for Python that allows users to create various types of static, animated, and interactive visualizations in Python. It provides a wide variety of customizable plots for different types of data, such as line plots, scatter plots, histograms, bar plots, pie charts, etc.

Matplotlib can be used for simple visualizations, as well as complex ones with multiple subplots and customizations. It is also compatible with other libraries like NumPy and Pandas, allowing users to easily create plots from data stored in arrays and dataframes.
In addition to its plotting capabilities, Matplotlib also offers a range of features for customizing the appearance of plots, including setting colors, line styles, marker styles, axis labels, legends, annotations, and more. With its flexibility and wide range of features, Matplotlib is a popular choice for data visualization in Python.

Seaborn:

Seaborn is a Python library for data visualization and is built on top of Matplotlib. It provides a high-level interface for creating informative and attractive statistical graphics. Seaborn makes it easy to create various types of plots such as scatterplots, line plots, bar plots, histograms, heatmaps, and many others.

Seaborn comes with several built-in datasets that can be used for practice and learning. It also provides various customization options such as color palettes, plot styles, and themes, making it easy to create publication-quality plots.

One of the main advantages of Seaborn is its ability to visualize complex statistical relationships in a clear and concise manner. For example, it can create scatterplots with linear regression lines, group bar plots with error bars, and many other types of plots that can provide insights into the data. Additionally, Seaborn integrates well with Pandas, making it easy to work with data in a DataFrame format.

Overall, Seaborn is a powerful tool for data visualization and can be used in various applications such as exploratory data analysis, data preprocessing, and model evaluation.

Scikit-learn:

Scikit-learn, commonly known as sklearn, is a popular machine learning library for Python. It provides a range of tools and algorithms for machine learning tasks such as classification, regression, clustering, and dimensionality reduction.

The library is built on top of other scientific computing packages, such as NumPy, SciPy, and matplotlib, and makes it easy to build and apply machine learning models through a unified and consistent API.
Sklearn includes a wide range of pre-processing, feature selection, and model selection tools to make machine learning tasks more efficient and accessible. It also includes utilities for data splitting, cross-validation, and model evaluation to help users to properly evaluate their models.

Sklearn is an open-source library and is widely used in industry and academia for a variety of machine learning applications.

TensorFlow:

TensorFlow is a popular open-source library for numerical computation and large-scale machine learning. It provides a flexible architecture for building various machine learning models, including artificial neural networks, convolutional neural networks, recurrent neural networks, and more. TensorFlow was developed by Google and is widely used in both academic and industry settings.

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow. It provides a user-friendly interface for building and training neural networks, making it easier for developers to create and experiment with different models. Keras allows users to quickly prototype and test new models without worrying about low-level details such as tensor manipulation, GPU acceleration, or weight initialization.

Imblearn:

imblearn is a Python library that provides various techniques for handling imbalanced datasets in machine learning. Imbalanced datasets occur when one class is significantly more frequent than the other in a binary classification problem, which can cause the model to be biased towards the majority class.

Imblearn provides various methods for handling imbalanced data, such as oversampling the minority class, undersampling the majority class, and generating synthetic samples using algorithms such as SMOTE (Synthetic Minority Over-sampling Technique). It also provides pipeline and ensemble methods for combining different techniques to improve the performance of the model.
Imblearn is built on top of scikit-learn, and can be easily integrated into any machine learning pipeline. It provides a user-friendly API and extensive documentation, making it easy for developers to implement these techniques in their projects.

Algorithms used

- Artificial Neural Network
- Logistic Regression
- KNN Classification
- Naive Bayes
- Support Vector Machines

Evaluation Metrics

Accuracy - In machine learning, accuracy is one of the most commonly used evaluation metrics to measure the performance of a classification model. Accuracy is the ratio of correctly predicted observations to the total number of observations in the dataset. It measures how well the model can correctly predict the class labels of the test data.

Precision - In the context of classification models, precision is a metric that measures the proportion of true positive predictions made by the model out of all the positive predictions it made. In other words, it measures the accuracy of positive predictions made by the model. Precision is calculated as the number of true positives divided by the sum of true positives and false positives.

Recall - In a classification report, recall is one of the key metrics used to evaluate the performance of a classification model. It provides information about the model's ability to correctly identify positive instances. A high recall score indicates that the model is able to correctly identify most of the positive instances, while a low recall score indicates that the model is missing many of the positive instances.
F1 Score - F1 score is a measure of a model's accuracy in binary classification tasks, which takes into account both precision and recall. It is the harmonic mean of precision and recall.

AUC ROC - AUC (Area Under the Curve) ROC (Receiver Operating Characteristic) is a metric used in classification tasks to evaluate the performance of a binary classification model. The AUC ROC metric measures the ability of a model to distinguish between the positive and negative classes.

The AUC ROC score ranges from 0 to 1, with a score of 0.5 indicating that the model is not better than random guessing, and a score of 1 indicating that the model is perfectly accurate. The higher the AUC ROC score, the better the model's ability to distinguish between the positive and negative classes.

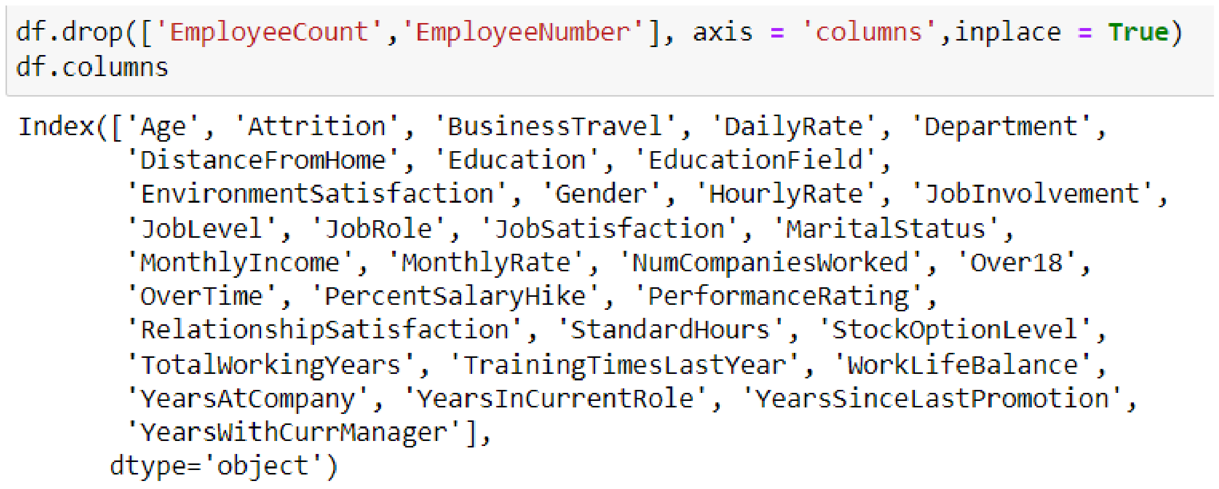
DATA ANALYSIS & MODELING

Data Preprocessing

For the smooth functioning of the model, it is important to feed it with a good and clean dataset. Thus, Data Preprocessing becomes an important step.
The data is observed and unnecessary features are removed. The categorical variables in the dataset should be converted to numerical form. Also scaling of the dataset is done to increase the performance of the model.
If the dataset is imbalanced, it is prone to make the model biased in prediction, hence it should be balanced before performing the model fit.

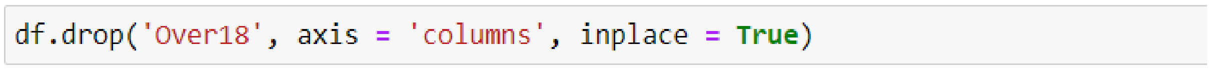
Data Cleaning -

A quick inspection of the dataset reveals that some features are the same for all employees such as *EmployeeCount* and *Over18* are omitted at this stage. Additionally, the *EmployeeNumber* feature has been omitted as its value is irrelevant to the classification problem.



```
df.drop(['EmployeeCount','EmployeeNumber'], axis = 'columns',inplace = True)
df.columns
```

```
Index(['Age', 'Attrition', 'BusinessTravel', 'DailyRate', 'Department',
       'DistanceFromHome', 'Education', 'EducationField',
       'EnvironmentSatisfaction', 'Gender', 'HourlyRate', 'JobInvolvement',
       'JobLevel', 'JobRole', 'JobSatisfaction', 'MaritalStatus',
       'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked', 'Over18',
       'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager'],
      dtype='object')
```



```
df.drop('Over18', axis = 'columns', inplace = True)
```

Categorical Data Encoding -

Some of the dataset's characteristics are categorical (nominal) values rather than numerical values.

Categorical characteristics cannot be used directly in most machine learning techniques. BusinessTravel, Department, EducationField, Gender, JobRole, MaritalStatus, and Overtime are all categorical characteristics in the original dataset. These characteristics must be numerically encoded.

To address this issue, one-hot encoding is utilized, in which the unique values and their corresponding numbers are identified first. Then, for each value, a one-shot binary vector is assigned. Gender, for example, which has two values (male and female), is converted into (1, 0) and (0, 1), respectively.

Yes and No are converted to 1 and 0 respectively. Other categorical variables are processed with pandas library to make dummy variables.

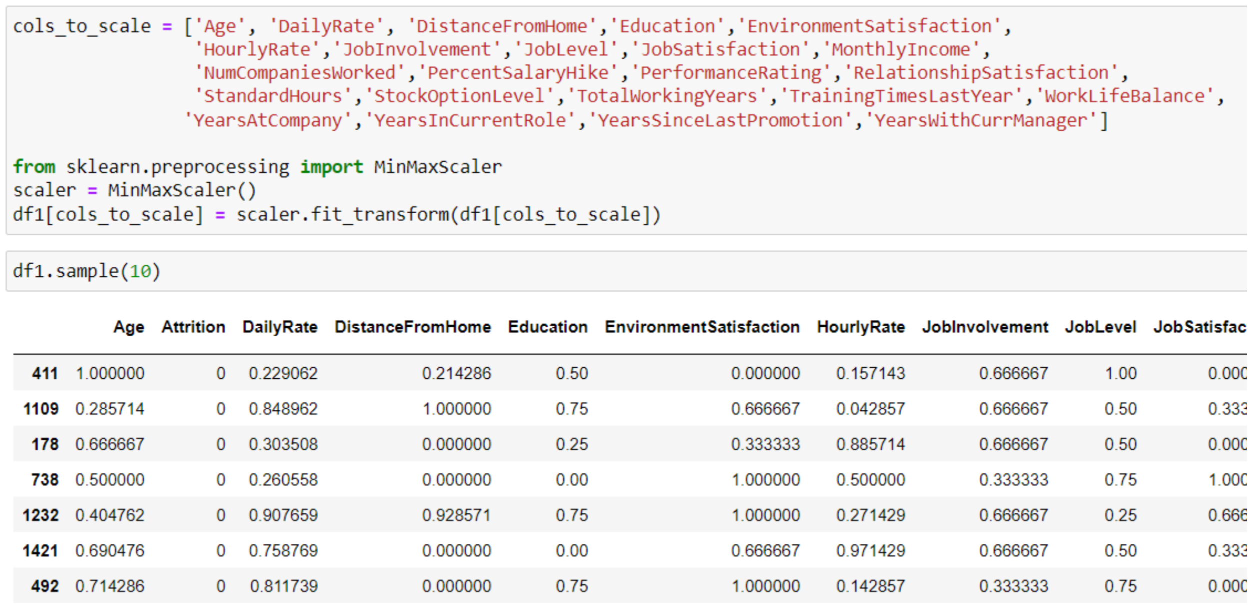
```
df['Attrition'] = df['Attrition'].replace({'Yes': 1, 'No': 0})
df['OverTime'] = df['OverTime'].replace({'Yes': 1, 'No': 0})
```

```
df1 = pd.get_dummies(data=df, columns = ['BusinessTravel','Department','EducationField',
                                         'Gender','JobRole','MaritalStatus'])
df1.columns
```

```
Index(['Age', 'Attrition', 'DailyRate', 'DistanceFromHome', 'Education',
       'EnvironmentSatisfaction', 'HourlyRate', 'JobInvolvement', 'JobLevel',
       'JobSatisfaction', 'MonthlyIncome', 'MonthlyRate', 'NumCompaniesWorked',
       'OverTime', 'PercentSalaryHike', 'PerformanceRating',
       'RelationshipSatisfaction', 'StandardHours', 'StockOptionLevel',
       'TotalWorkingYears', 'TrainingTimesLastYear', 'WorkLifeBalance',
       'YearsAtCompany', 'YearsInCurrentRole', 'YearsSinceLastPromotion',
       'YearsWithCurrManager', 'BusinessTravel_Non-Travel',
       'BusinessTravel_Travel_Frequently', 'BusinessTravel_Travel_Rarely',
       'Department_Human Resources', 'Department_Research & Development',
       'Department_Sales', 'EducationField_Human Resources',
       'EducationField_Life Sciences', 'EducationField_Marketing',
       'EducationField_Medical', 'EducationField_Other',
       'EducationField_Technical Degree', 'Gender_Female', 'Gender_Male',
       'JobRole_Healthcare Representative', 'JobRole_Human Resources',
       'JobRole_Laboratory Technician', 'JobRole_Manager',
       'JobRole_Manufacturing Director', 'JobRole_Research Director',
       'JobRole_Research Scientist', 'JobRole_Sales Executive',
       'JobRole_Sales Representative', 'MaritalStatus_Divorced',
       'MaritalStatus_Married', 'MaritalStatus_Single'],
      dtype='object')
```

Rescaling -

The ranges of features vary widely, resulting in poor classification results since features with broad ranges may be given more weight than other features. To solve this problem, we must rescale feature values to be in the same range. Normalisation is a typical method of rescaling feature values, in which values are rescaled to a given time. The feature values in this work are rescaled to the range [0, 1].
The MinMaxScaler function in Sklearn library is used to scale the variables.



```
cols_to_scale = ['Age', 'DailyRate', 'DistanceFromHome','Education','EnvironmentSatisfaction',
                 'HourlyRate','JobInvolvement','JobLevel','JobSatisfaction','MonthlyIncome',
                 'NumCompaniesWorked','PercentSalaryHike','PerformanceRating','RelationshipSatisfaction',
                 'StandardHours','StockOptionLevel','TotalWorkingYears','TrainingTimesLastYear','WorkLifeBalance',
                'YearsAtCompany','YearsInCurrentRole','YearsSinceLastPromotion','YearsWithCurrManager']

from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
df1[cols_to_scale] = scaler.fit_transform(df1[cols_to_scale])
```

```
df1.sample(10)
```

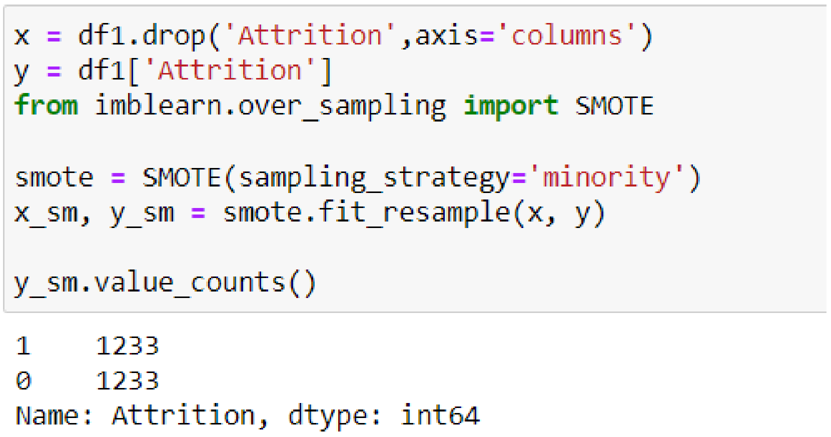
```
           Age  Attrition  DailyRate  DistanceFromHome  Education  EnvironmentSatisfaction  HourlyRate  JobInvolvement  JobLevel  JobSatisfac
 411  1.000000          0   0.229062          0.214286       0.50                 0.000000    0.157143        0.666667      1.00        0.000
1109  0.285714          0   0.848962          1.000000       0.75                 0.666667    0.042857        0.666667      0.50        0.333
 178  0.666667          0   0.303508          0.000000       0.25                 0.333333    0.885714        0.666667      0.50        0.000
 738  0.500000          0   0.260558          0.000000       0.00                 1.000000    0.500000        0.333333      0.75        1.000
1232  0.404762          0   0.907659          0.928571       0.75                 1.000000    0.271429        0.666667      0.25        0.666
1421  0.690476          0   0.758769          0.000000       0.00                 0.666667    0.971429        0.666667      0.50        0.333
 492  0.714286          0   0.811739          0.000000       0.75                 1.000000    0.142857        0.333333      0.75        0.000
```

Here we can see that all the values are scaled to a value between 0 to 1.

Dataset Balancing

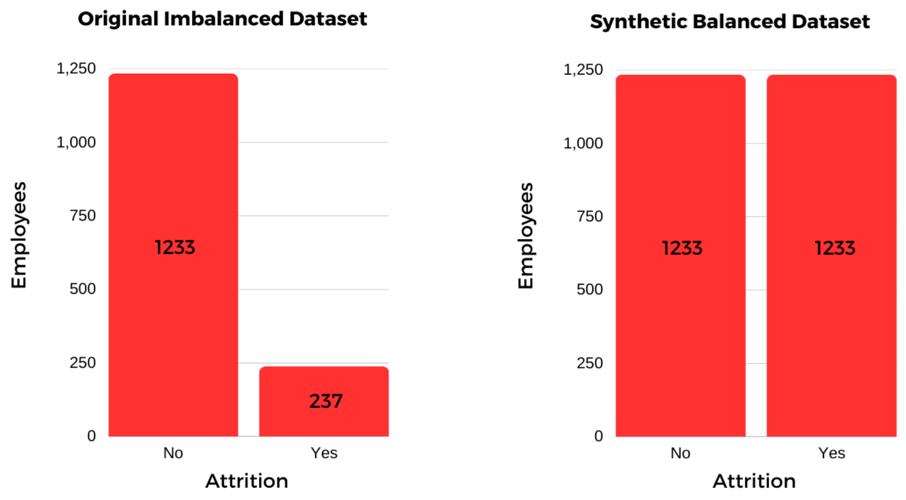
The IBM Watson dataset is not a balanced one. If we check the count of ‘Yes’ and ‘No’ under the target variable ‘Attrition’, it is 237 and 1233 respectively. This makes the class ‘No’ dominant and thus brings bias to the prediction. In order to overcome this problem, there are certain techniques to adopt such as oversampling the minority, undersampling the majority, Synthetic Minority Oversampling TEchnique (SMOTE) .etc.
Here we have used SMOTE to oversample the minority class. Thus the dataset is balanced and chances for bias are minimized.

SMOTE - SMOTE (Synthetic Minority Over-sampling Technique) is a widely used oversampling technique in machine learning for balancing imbalanced datasets. It works by taking each sample of the minority class and inserting synthetic examples along the line segments connecting the minority class sample to its k nearest neighbors in feature space to generate synthetic samples of the minority class. This method generates additional data points that are comparable to the minority class, enhancing the dataset's representation of the minority class. To control the degree of oversampling, change the value of k. To balance the data and increase the performance of machine learning models, SMOTE is frequently used in conjunction with undersampling techniques such as random undersampling.



```
x = df1.drop('Attrition',axis='columns')
y = df1['Attrition']
from imblearn.over_sampling import SMOTE

smote = SMOTE(sampling_strategy='minority')
x_sm, y_sm = smote.fit_resample(x, y)

y_sm.value_counts()
```

```
1    1233
0    1233
Name: Attrition, dtype: int64
```

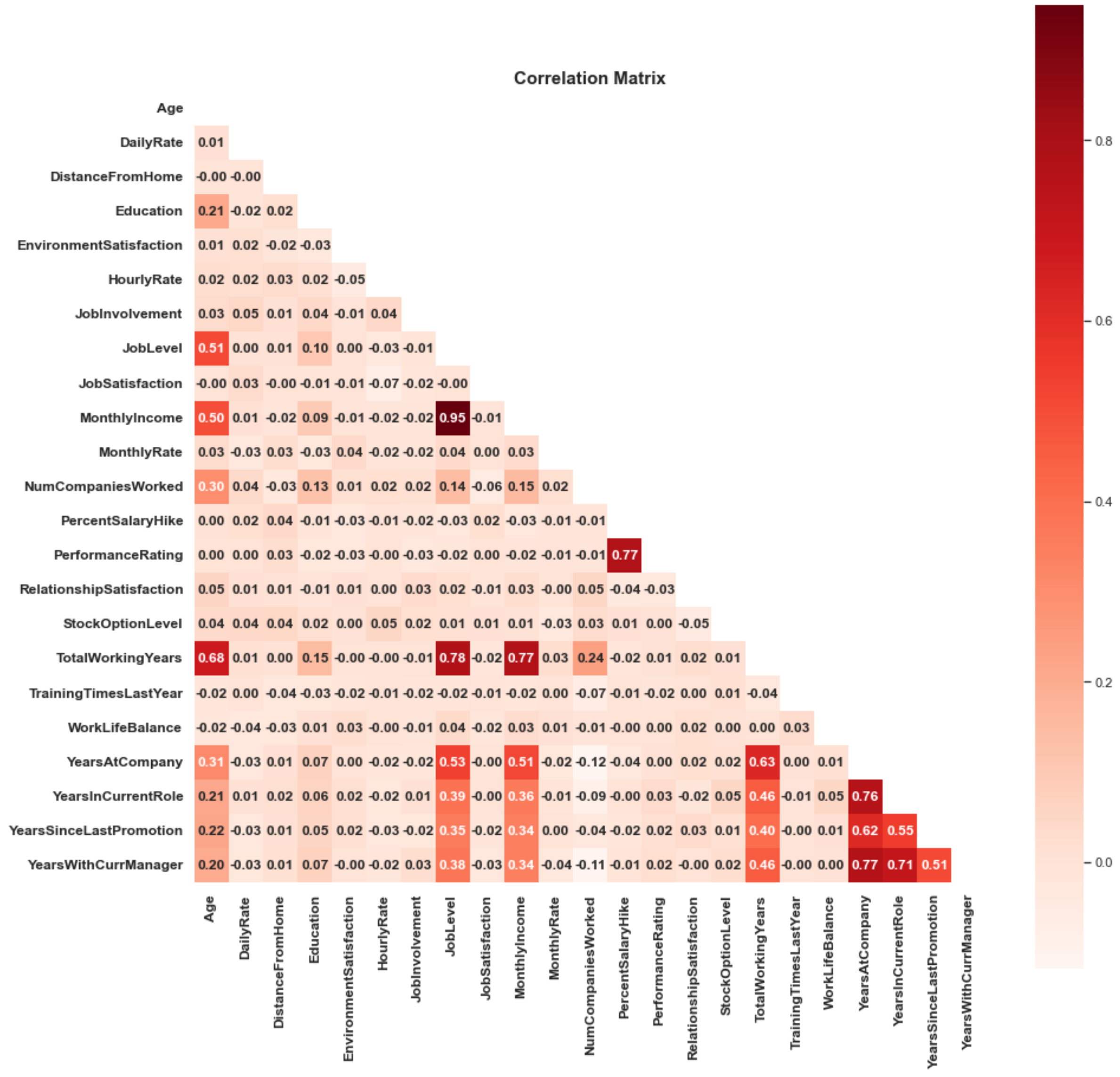


Before and After the balancing of dataset

Here we have made both (Attrition = Yes) and (Attrition = No) to have similar counts. SMOTE makes synthetic data points instead of duplicating the already existing data points, hence reducing the problem overfitting as well.

Exploratory Data Analysis

The correlation matrix is commonly used to comprehend the link between the dataset's properties. The correlation matrix of our dataset is depicted in the figure below. The cell colors range from white to crimson. Pale red cells indicate no correlation, while red variations indicate a substantial association. White variations indicate a negative association between dataset properties.

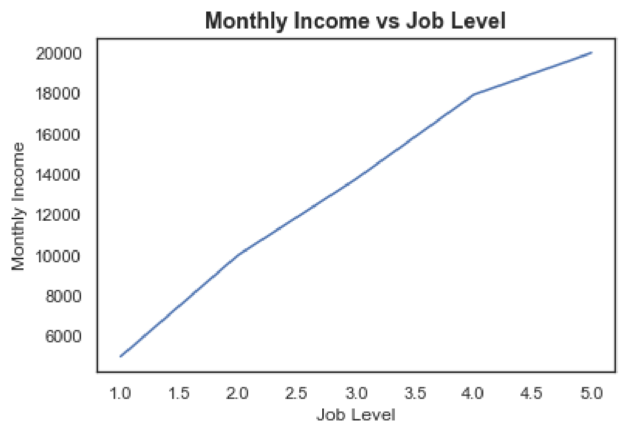


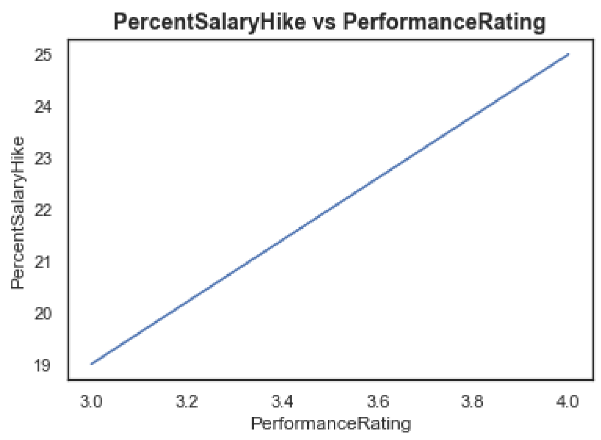
	Age	DailyRate	DistanceFromHome	Education	EnvironmentSatisfaction	HourlyRate	JobInvolvement	JobLevel	JobSatisfaction	MonthlyIncome	MonthlyRate	NumCompaniesWorked	PercentSalaryHike	PerformanceRating	RelationshipSatisfaction	StockOptionLevel	TotalWorkingYears	TrainingTimesLastYear	WorkLifeBalance	YearsAtCompany	YearsInCurrentRole	YearsSinceLastPromotion	YearsWithCurrManager
Age																							
DailyRate	0.01																						
DistanceFromHome	-0.00	-0.00																					
Education	0.21	-0.02	0.02																				
EnvironmentSatisfaction	0.01	0.02	-0.02	-0.03																			
HourlyRate	0.02	0.02	0.03	0.02	-0.05																		
JobInvolvement	0.03	0.05	0.01	0.04	-0.01	0.04																	
JobLevel	0.51	0.00	0.01	0.10	0.00	-0.03	-0.01																
JobSatisfaction	-0.00	0.03	-0.00	-0.01	-0.01	-0.07	-0.02	-0.00															
MonthlyIncome	0.50	0.01	-0.02	0.09	-0.01	-0.02	-0.02	0.95	-0.01														
MonthlyRate	0.03	-0.03	0.03	-0.03	0.04	-0.02	-0.02	0.04	0.00	0.03													
NumCompaniesWorked	0.30	0.04	-0.03	0.13	0.01	0.02	0.02	0.14	-0.06	0.15	0.02												
PercentSalaryHike	0.00	0.02	0.04	-0.01	-0.03	-0.01	-0.02	-0.03	0.02	-0.03	-0.01	-0.01											
PerformanceRating	0.00	0.00	0.03	-0.02	-0.03	-0.00	-0.03	-0.02	0.00	-0.02	-0.01	-0.01	0.77										
RelationshipSatisfaction	0.05	0.01	0.01	-0.01	0.01	0.00	0.03	0.02	-0.01	0.03	-0.00	0.05	-0.04	-0.03									
StockOptionLevel	0.04	0.04	0.04	0.02	0.00	0.05	0.02	0.01	0.01	0.01	-0.03	0.03	0.01	0.00	-0.05								
TotalWorkingYears	0.68	0.01	0.00	0.15	-0.00	-0.00	-0.01	0.78	-0.02	0.77	0.03	0.24	-0.02	0.01	0.02	0.01							
TrainingTimesLastYear	-0.02	0.00	-0.04	-0.03	-0.02	-0.01	-0.02	-0.02	-0.01	-0.02	0.00	-0.07	-0.01	-0.02	0.00	0.01	-0.04						
WorkLifeBalance	-0.02	-0.04	-0.03	0.01	0.03	-0.00	-0.01	0.04	-0.02	0.03	0.01	-0.01	-0.00	0.00	0.02	0.00	0.00	0.03					
YearsAtCompany	0.31	-0.03	0.01	0.07	0.00	-0.02	-0.02	0.53	-0.00	0.51	-0.02	-0.12	-0.04	0.00	0.02	0.02	0.63	0.00	0.01				
YearsInCurrentRole	0.21	0.01	0.02	0.06	0.02	-0.02	0.01	0.39	-0.00	0.36	-0.01	-0.09	-0.00	0.03	-0.02	0.05	0.46	-0.01	0.05	0.76			
YearsSinceLastPromotion	0.22	-0.03	0.01	0.05	0.02	-0.03	-0.02	0.35	-0.02	0.34	0.00	-0.04	-0.02	0.02	0.03	0.01	0.40	-0.00	0.01	0.62	0.55		
YearsWithCurrManager	0.20	-0.03	0.01	0.07	-0.00	-0.02	0.03	0.38	-0.03	0.34	-0.04	-0.11	-0.01	0.02	-0.00	0.02	0.46	-0.00	0.00	0.77	0.71	0.51	

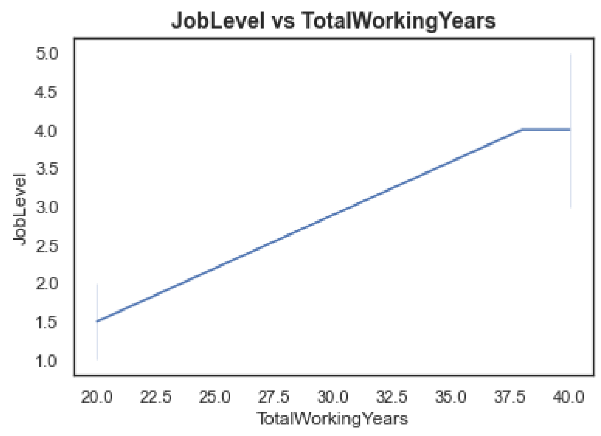
By analyzing the correlation matrix, we observe the following findings:

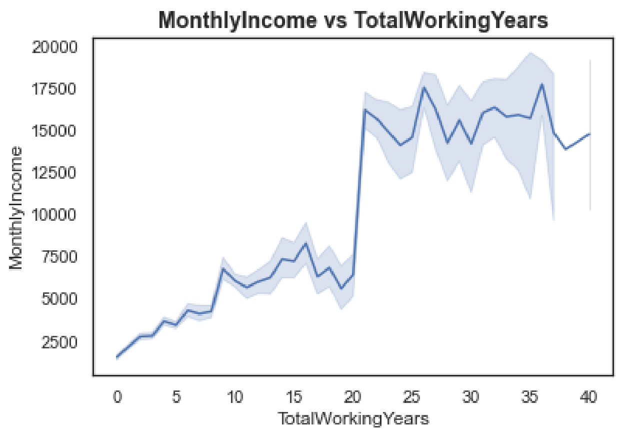
• "MonthlyIncome" is highly correlated with the "JobLevel".
• "PerformanceRating" is correlated with "PercentSalaryHike".
• "TotalWorkingYears" is correlated with "JobLevel" and "MonthlyIncome".
• "YearsAtCompany" is correlated with "YearsInCurrentRole" and "YearsWithCurrentManager".
• "TotalWorkingYears" is correlated with "Age".

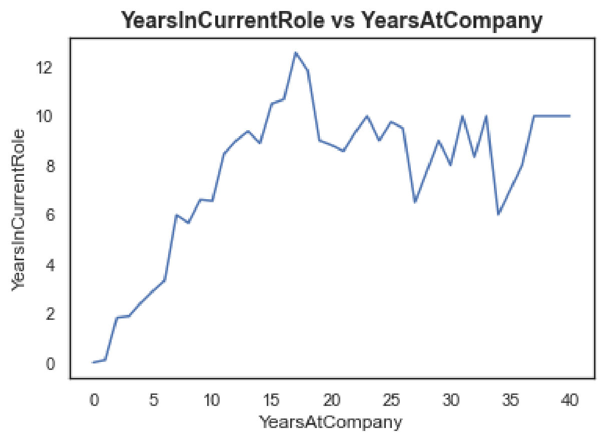
- “YeasAtCompany” is moderately correlated with “YearsSinceLastPromotion” and “TotalWorkingYears”.

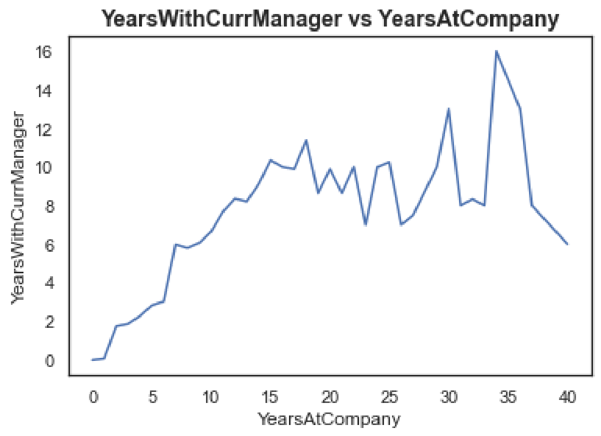


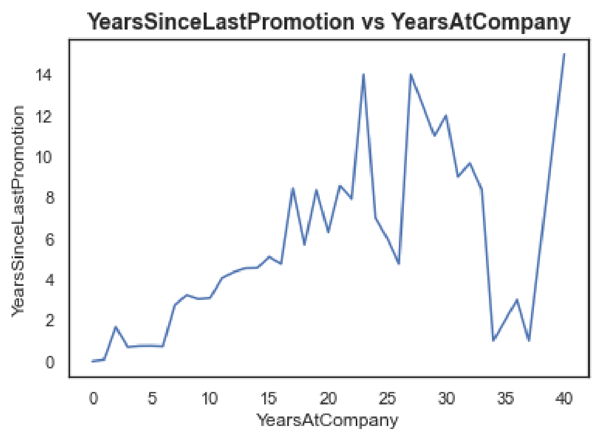


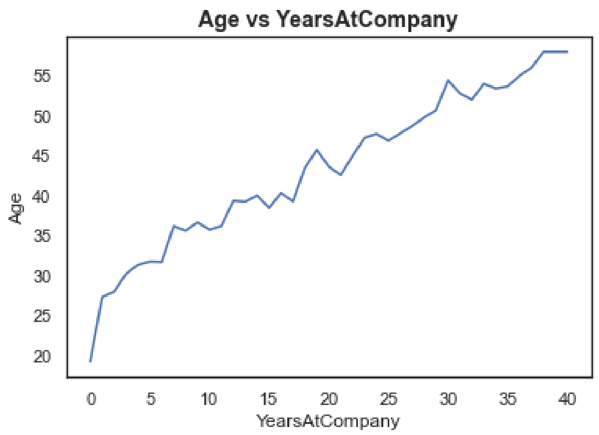


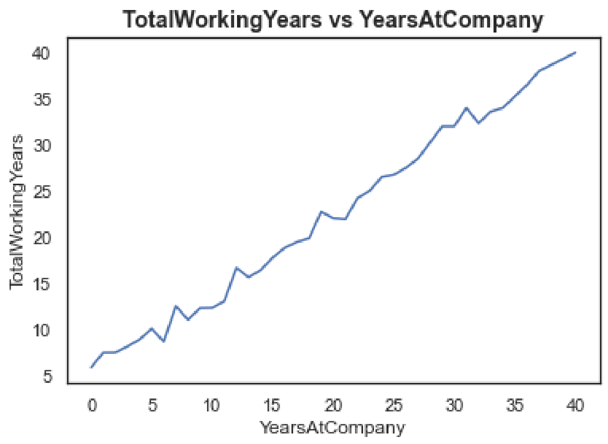








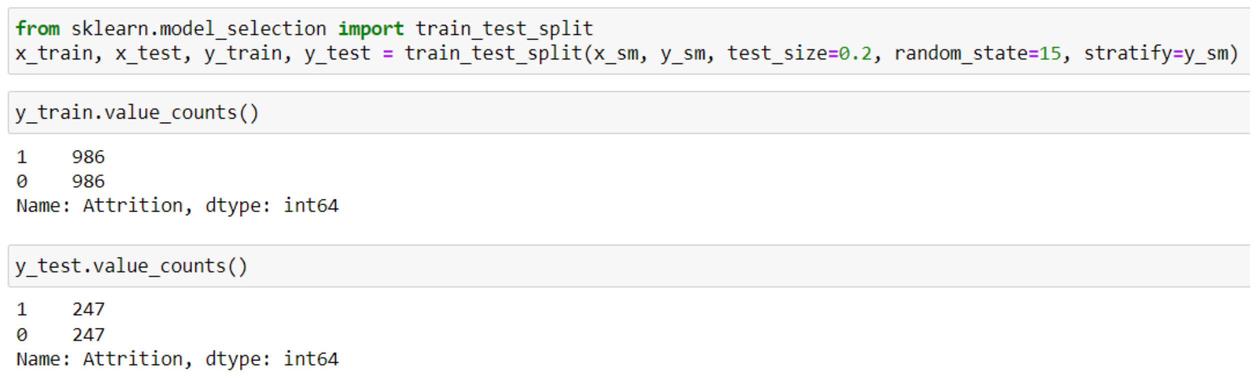




Correlation graphs

Splitting of dataset into test and train

The balanced dataset is then split into test and train datasets with the ratio of 80% : 20%. Here 80% of the dataset will be used to train the models and the remaining 20% is used to test and evaluate the performance of the model.
The Test Train Split function in the Sklearn module is used to do this task.



```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_sm, y_sm, test_size=0.2, random_state=15, stratify=y_sm)
```

```
y_train.value_counts()
```

```
1    986
0    986
Name: Attrition, dtype: int64
```

```
y_test.value_counts()
```

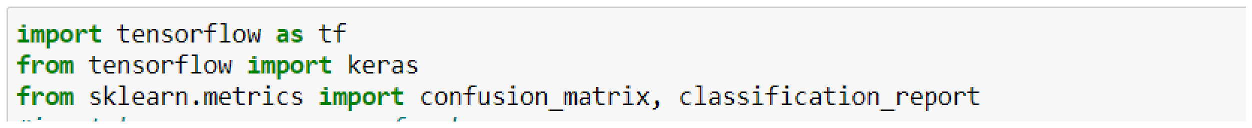
```
1    247
0    247
Name: Attrition, dtype: int64
```

The x_test and y_test datasets are fitted into the models to train it.

Prediction Models

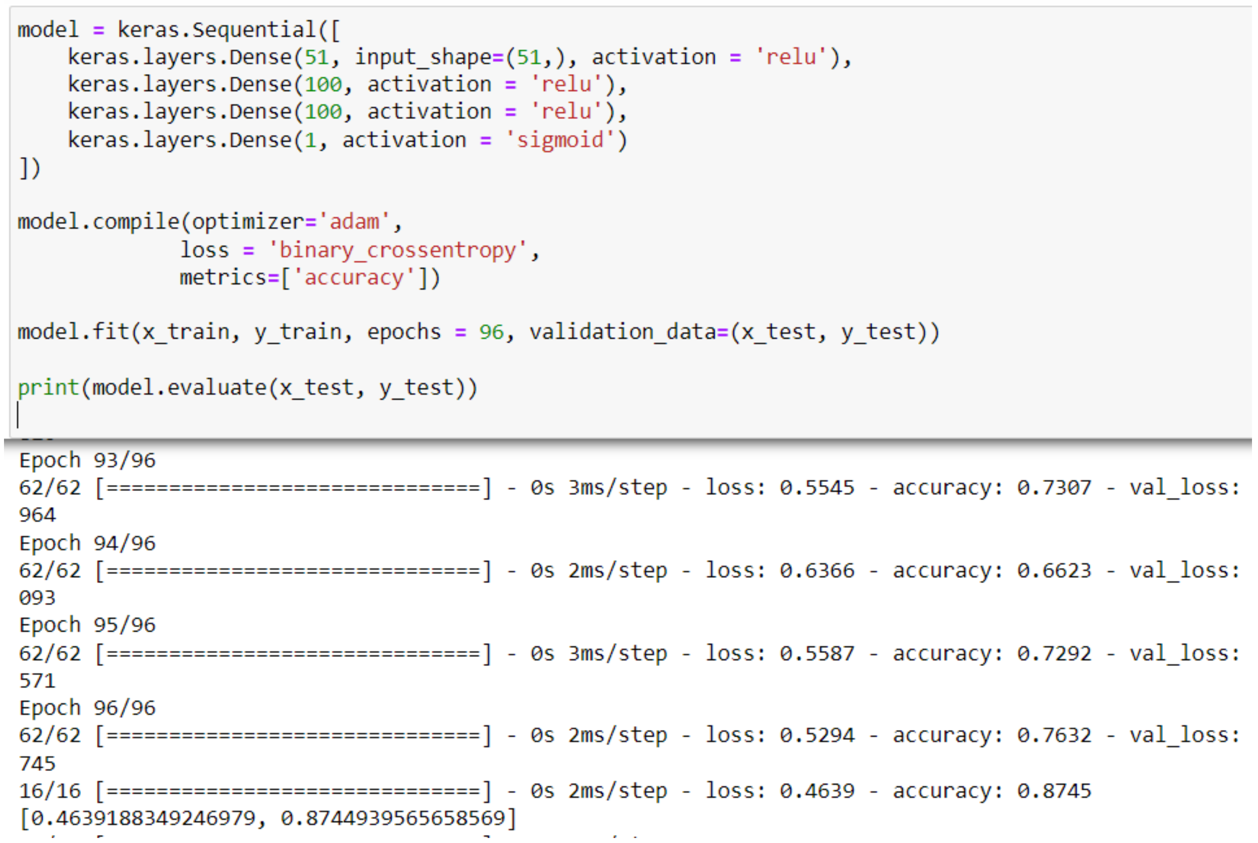
Artificial Neural Network -

Keras in Tensorflow Library is used to make the ANN model. The model is then trained with the train datasets and evaluated. Using different numbers of nodes, hidden layers and epochs the model is created and the parameters which give best results are chosen to be in the final model.
Here the model has 51 input nodes (51 Features of the Dataset) and two hidden layers with 100 nodes each with activation function ‘Relu’. The output layer has 1 node with activation function ‘Sigmoid’ as it is a binary classification model. The loss function used here is ‘Binary cross entropy and ‘Adam’ optimizer.
Adam optimizer is a stochastic gradient descent optimization algorithm that is commonly used for training deep neural networks. It is an adaptive learning rate optimization algorithm that calculates individual adaptive learning rates for each parameter by computing the exponentially weighted averages of past gradients and past squared gradients.



```
import tensorflow as tf
from tensorflow import keras
from sklearn.metrics import confusion_matrix, classification_report
```

Importing the required libraries



```
model = keras.Sequential([
    keras.layers.Dense(51, input_shape=(51,), activation = 'relu'),
    keras.layers.Dense(100, activation = 'relu'),
    keras.layers.Dense(100, activation = 'relu'),
    keras.layers.Dense(1, activation = 'sigmoid')
])

model.compile(optimizer='adam',
             loss = 'binary_crossentropy',
             metrics=['accuracy'])

model.fit(x_train, y_train, epochs = 96, validation_data=(x_test, y_test))

print(model.evaluate(x_test, y_test))
```

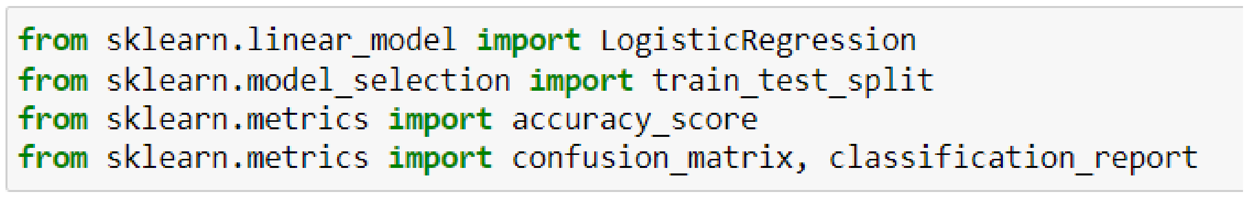
```
Epoch 93/96
62/62 [==============================] - 0s 3ms/step - loss: 0.5545 - accuracy: 0.7307 - val_loss:
964
Epoch 94/96
62/62 [==============================] - 0s 2ms/step - loss: 0.6366 - accuracy: 0.6623 - val_loss:
093
Epoch 95/96
62/62 [==============================] - 0s 3ms/step - loss: 0.5587 - accuracy: 0.7292 - val_loss:
571
Epoch 96/96
62/62 [==============================] - 0s 2ms/step - loss: 0.5294 - accuracy: 0.7632 - val_loss:
745
16/16 [==============================] - 0s 2ms/step - loss: 0.4639 - accuracy: 0.8745
[0.4639188349246979, 0.8744939565658569]
```

Training and evaluating the loss and accuracy of the model

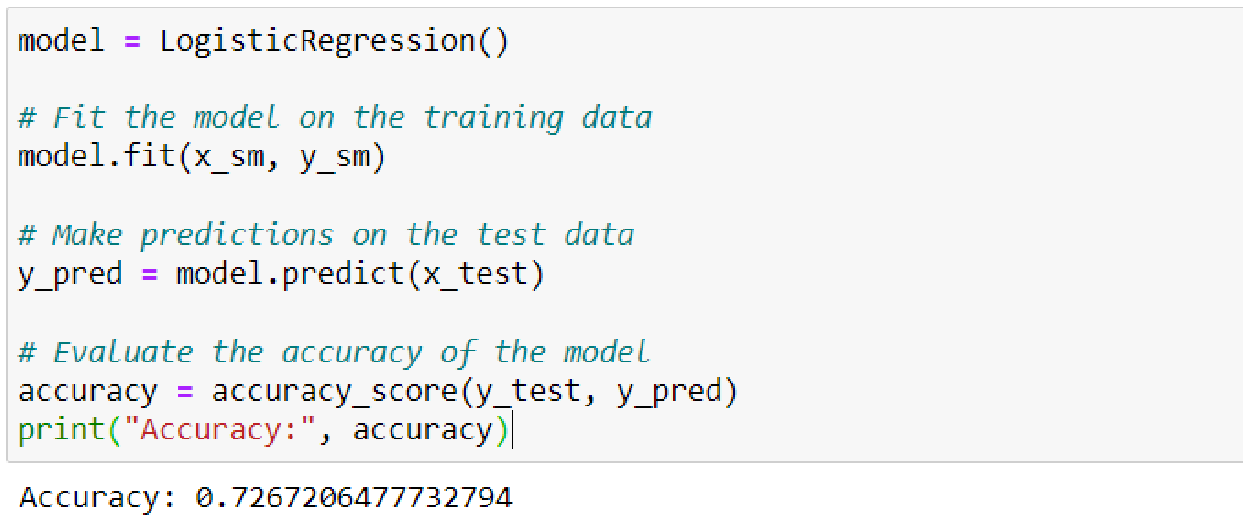
Observation :- Here the model is showing an accuracy of **87.45%** with a loss of **0.4639**. So this model can be finalized as its good.

Logistic Regression –

Logistic regression in the Sklearn library is used to create the model.
The balanced test datasets are fitted into the model and its performance is evaluated.



```python
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn.metrics import confusion_matrix, classification_report
```



```python
model = LogisticRegression()

# Fit the model on the training data
model.fit(x_sm, y_sm)

# Make predictions on the test data
y_pred = model.predict(x_test)

# Evaluate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7267206477732794
```

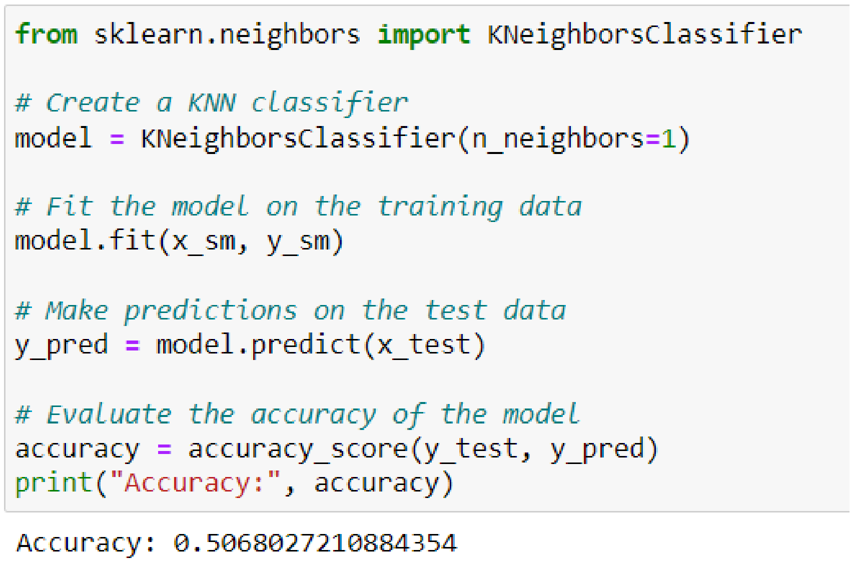
Observation :- The model returned an accuracy of **72.67%.**

KNN Classification -

From Sklearn Library, KNeighborsClassifier is imported and used to create a model for classification. Initially the model is run with K = 1 and the accuracy is noted and again the model is run with K = 3 and observed for any improvement in accuracy.

k represents the number of neighboring data points considered for making a prediction for a new data point.
When k=1, only the closest data point is considered for making the prediction. This can result in a model that overfits the training data and may not generalize well to new, unseen data. When k=3, the model considers the three closest data points for making a prediction.

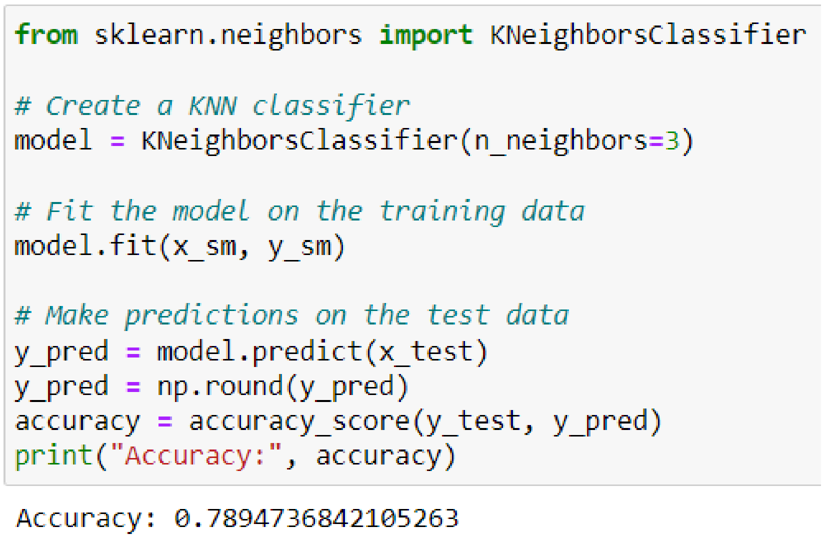
K =1



```
from sklearn.neighbors import KNeighborsClassifier

# Create a KNN classifier
model = KNeighborsClassifier(n_neighbors=1)

# Fit the model on the training data
model.fit(x_sm, y_sm)

# Make predictions on the test data
y_pred = model.predict(x_test)

# Evaluate the accuracy of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.5068027210884354
```

K = 3



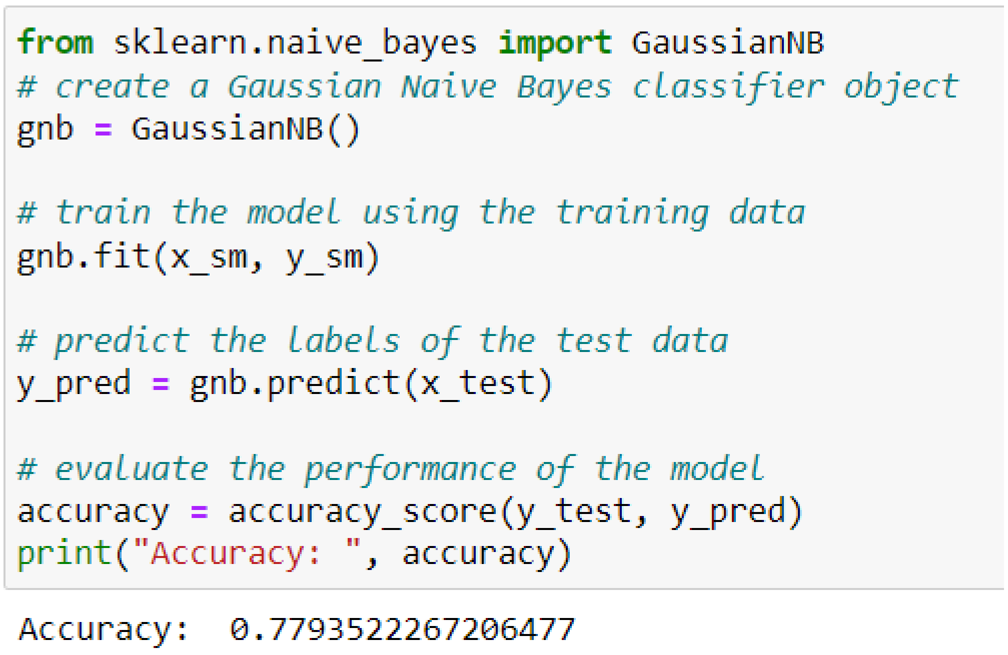
```
from sklearn.neighbors import KNeighborsClassifier

# Create a KNN classifier
model = KNeighborsClassifier(n_neighbors=3)

# Fit the model on the training data
model.fit(x_sm, y_sm)

# Make predictions on the test data
y_pred = model.predict(x_test)
y_pred = np.round(y_pred)
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7894736842105263
```

Observation :- With **K = 1** the model is having an accuracy of just **50.68%**
And with **K = 3** the model performs comparatively better with an accuracy of **78.95%**.

Naive Bayes -

GaussianNB in the Sklearn library is used to build a classification model.
The model is trained with the balanced train datasets and is evaluated for accuracy.



```
from sklearn.naive_bayes import GaussianNB
# create a Gaussian Naive Bayes classifier object
gnb = GaussianNB()

# train the model using the training data
gnb.fit(x_sm, y_sm)

# predict the labels of the test data
y_pred = gnb.predict(x_test)

# evaluate the performance of the model
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy: ", accuracy)
```

```
Accuracy:  0.7793522267206477
```

Observation :-

The NB model has an accuracy of **77.93%.**

Support Vector Machine (SVM) -

Using svm in the Sklearn library, the model is built with a linear kernel and various Non Linear kernels such as ‘rbf’ and ‘polynomial’.
Their individual accuracies are noted down and compared with each other.

Linear

```
from sklearn import svm

clf = svm.SVC(kernel='linear')

clf.fit(x_sm, y_sm)

y_pred = clf.predict(x_test)

# Evaluate the performance of the model
print('Accuracy:', accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.708502024291498
```

Gaussian SVM

```
clf = svm.SVC(kernel='rbf')

clf.fit(x_sm, y_sm)

y_pred = clf.predict(x_test)

# Evaluate the performance of the model
print('Accuracy:', accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.46963562753036436
```

Cubic SVM

```
clf = svm.SVC(kernel='poly', degree = 3)

clf.fit(x_sm, y_sm)

y_pred = clf.predict(x_test)

# Evaluate the performance of the model
print('Accuracy:', accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.5040485829959515
```

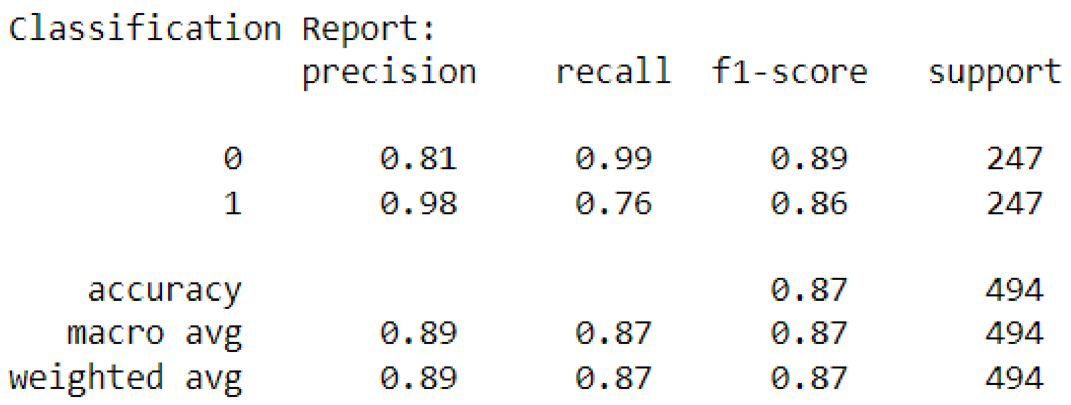
Observation :-

SVM Model	Accuracy
Linear SVM	**70.85%**
Gaussian SVM	**46.96%**
Cubic SVM	**50.40%**

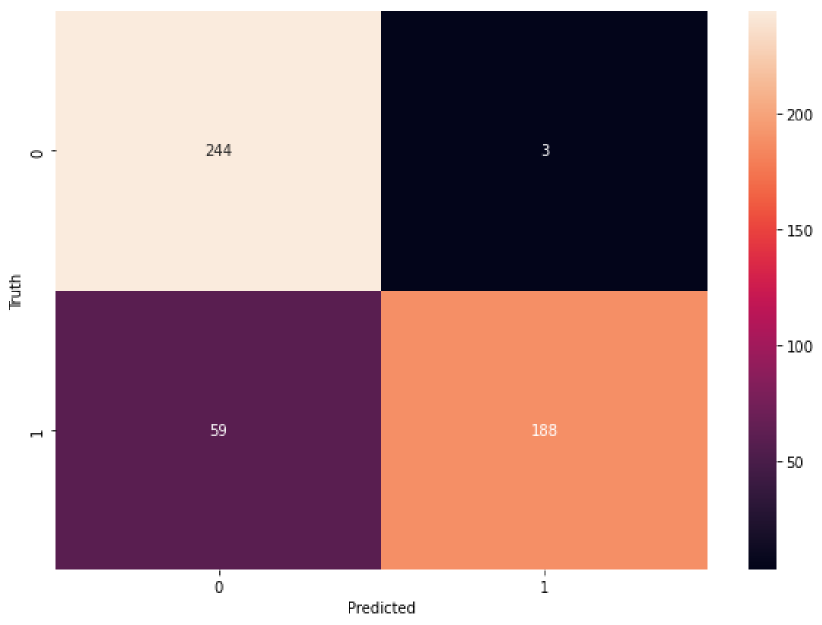
Evaluation of the Models

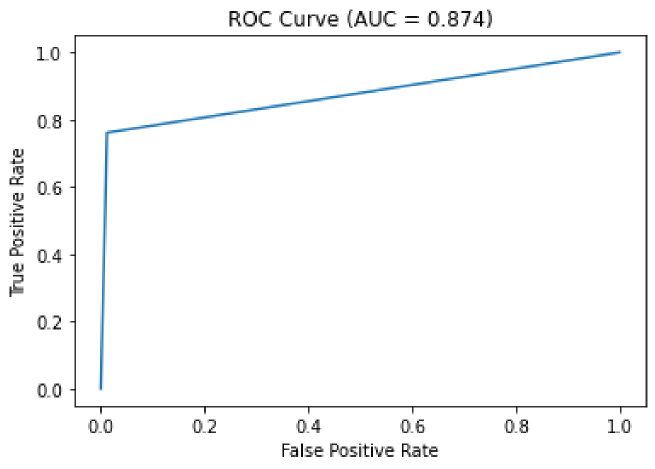
All the models are evaluated by comparing their **Classification Reports** which contain various metrics such as **Accuracy, Precision, Recall and F1 Score**. The **confusion matrix** for each model is obtained for evaluation.
Along with that the **ROC** curve of each model is plotted and the **AUC** score is extracted to further compare the performance.

ANN -



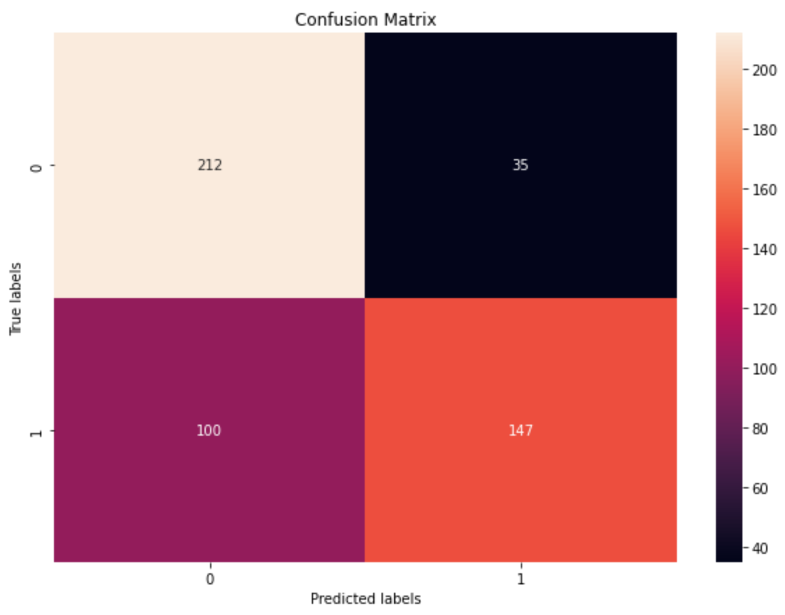
```
Classification Report:
              precision    recall  f1-score   support

           0       0.81      0.99      0.89       247
           1       0.98      0.76      0.86       247

    accuracy                           0.87       494
   macro avg       0.89      0.87      0.87       494
weighted avg       0.89      0.87      0.87       494
```

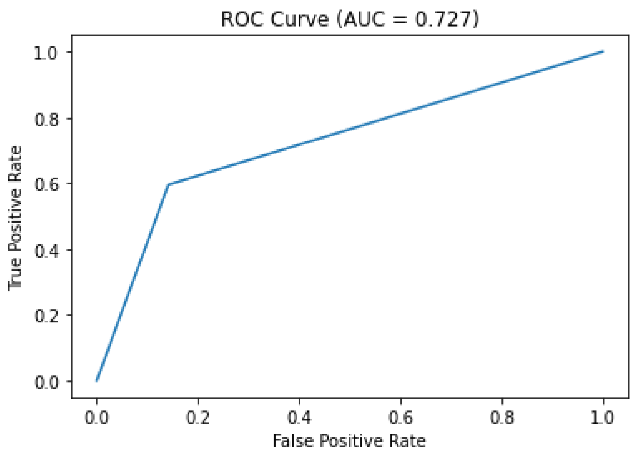




Logistic Regression -

```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.68      0.86      0.76       247
         1.0       0.81      0.60      0.69       247

    accuracy                           0.73       494
   macro avg       0.74      0.73      0.72       494
weighted avg       0.74      0.73      0.72       494
```

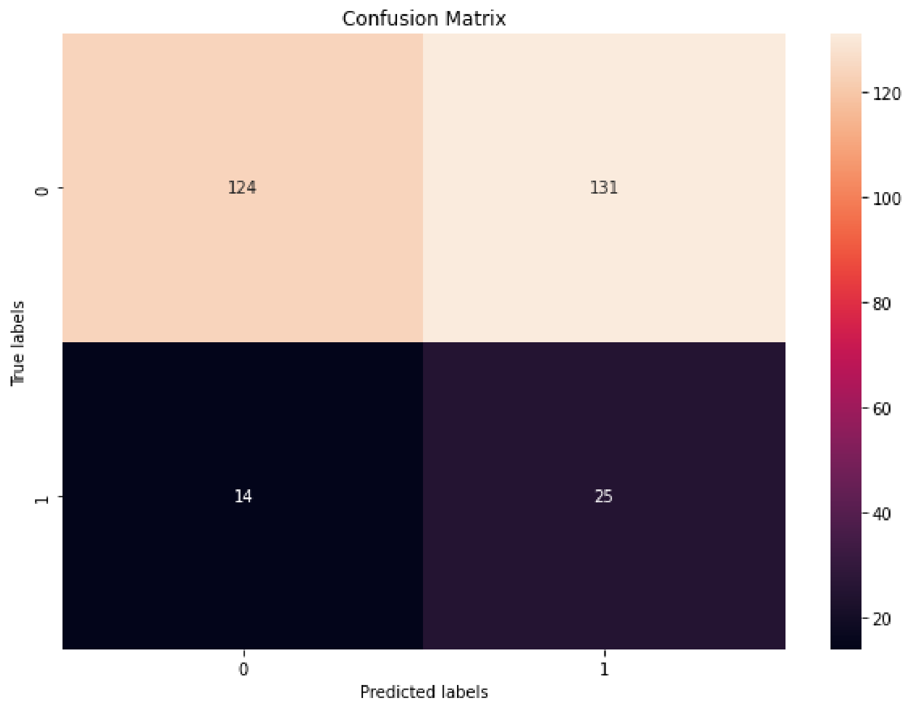


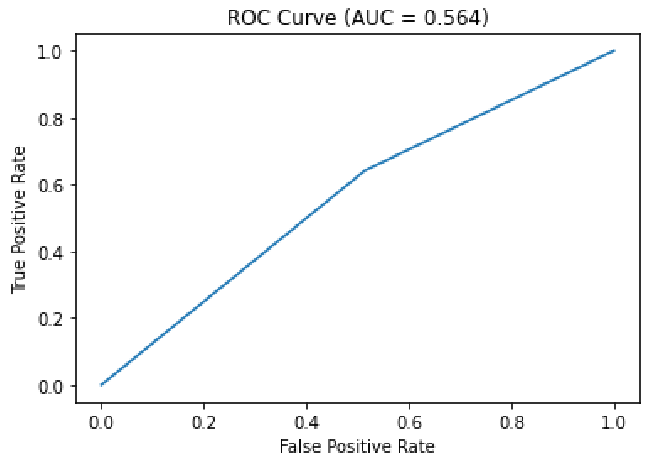


KNN :-

K= 1

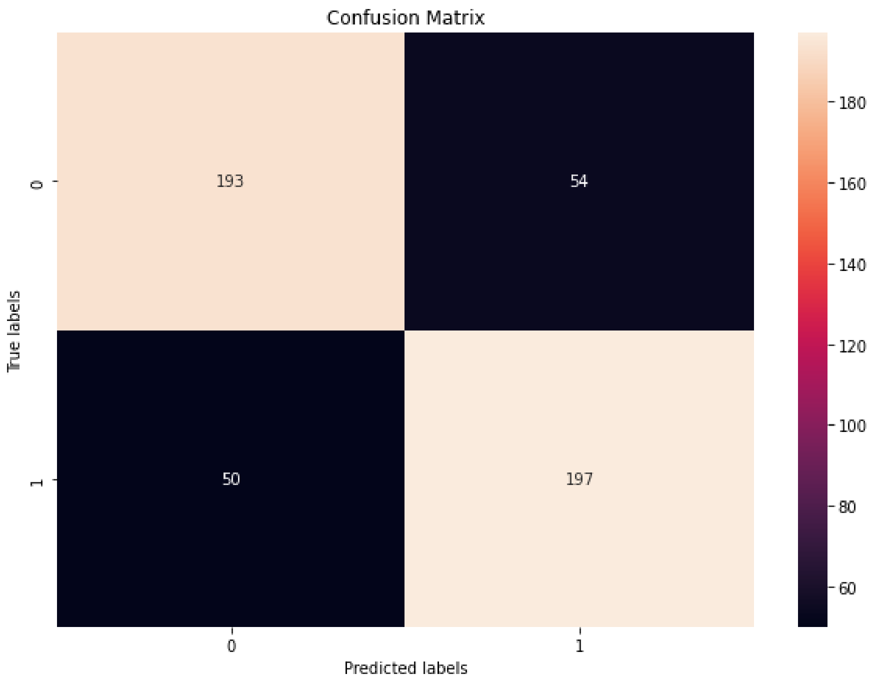
```
              precision    recall  f1-score   support

           0       0.90      0.49      0.63       255
           1       0.16      0.64      0.26        39

    accuracy                           0.51       294
   macro avg       0.53      0.56      0.44       294
weighted avg       0.80      0.51      0.58       294
```

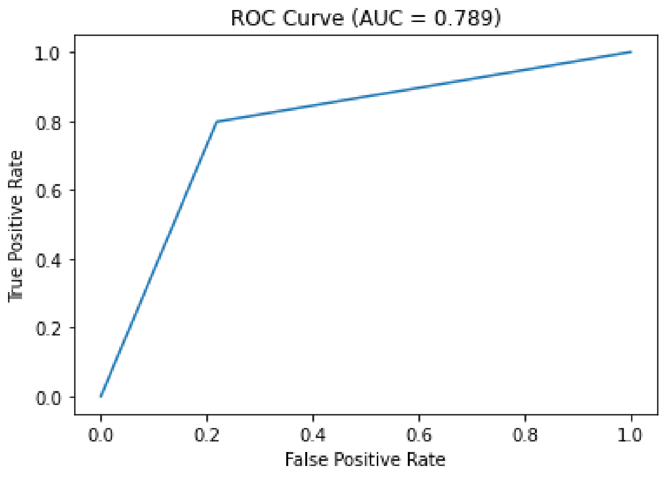




K = 3

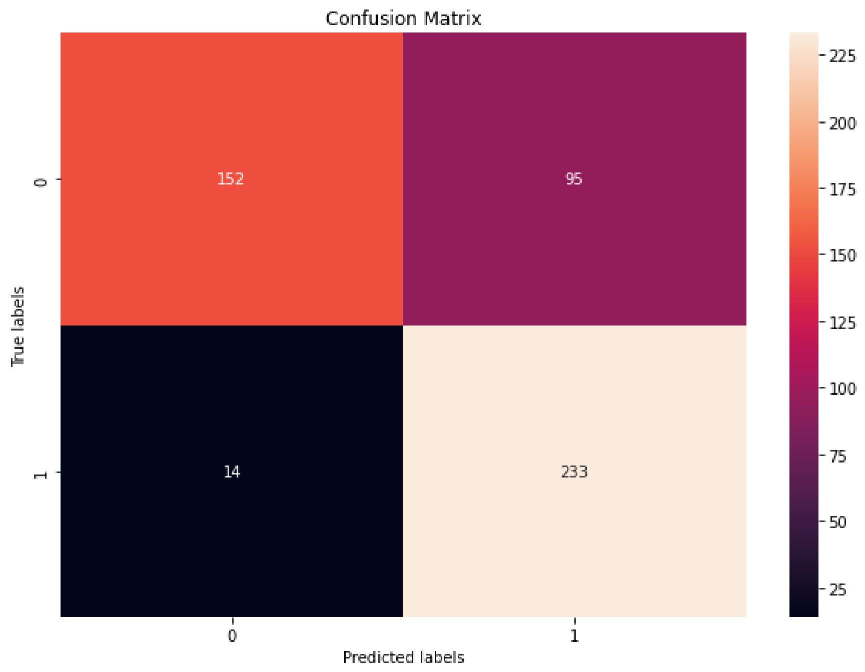
```
Classification Report:
               precision    recall  f1-score   support

         0.0       0.79      0.78      0.79       247
         1.0       0.78      0.80      0.79       247

    accuracy                           0.79       494
   macro avg       0.79      0.79      0.79       494
weighted avg       0.79      0.79      0.79       494
```

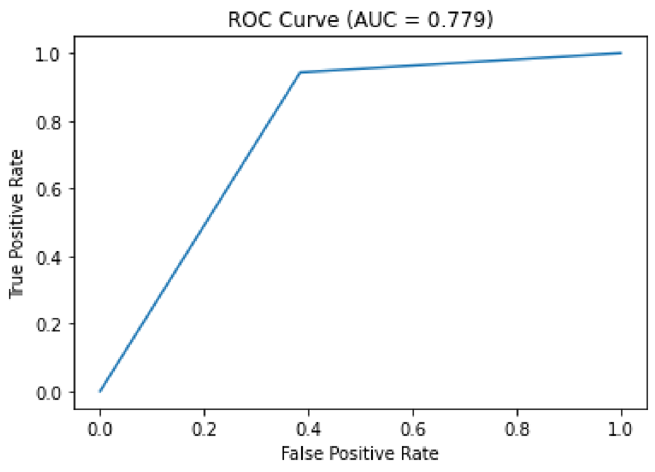




Naive Bayes -

```
Classification Report:
              precision    recall  f1-score   support

         0.0       0.92      0.62      0.74       247
         1.0       0.71      0.94      0.81       247

    accuracy                           0.78       494
   macro avg       0.81      0.78      0.77       494
weighted avg       0.81      0.78      0.77       494
```

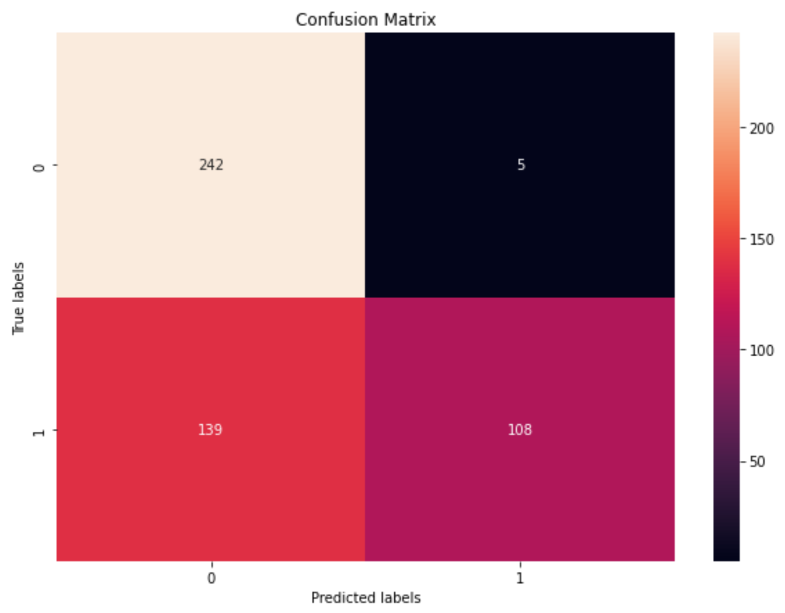


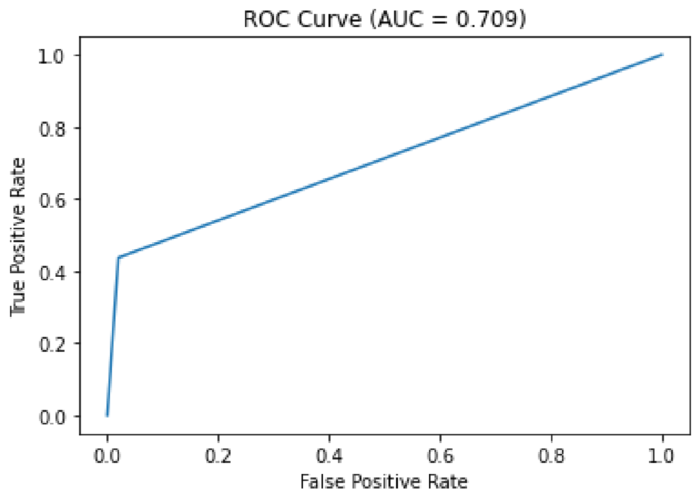


Support Vector Machine (SVM) -

Linear

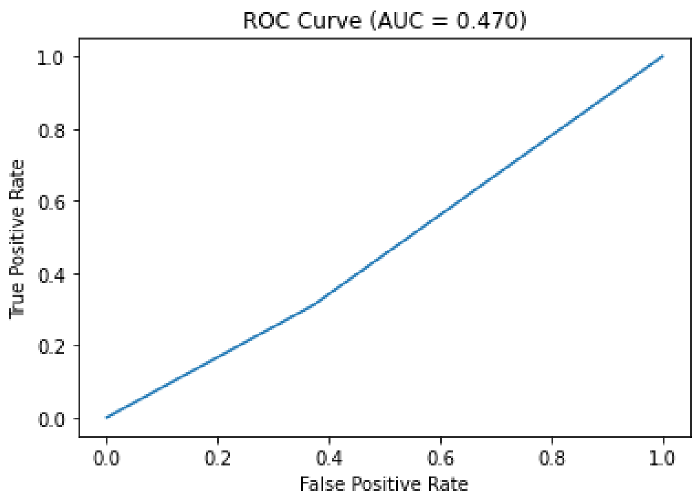
```
Classification report:
              precision    recall  f1-score   support

         0.0       0.64      0.98      0.77       247
         1.0       0.96      0.44      0.60       247

    accuracy                           0.71       494
   macro avg       0.80      0.71      0.69       494
weighted avg       0.80      0.71      0.69       494
```





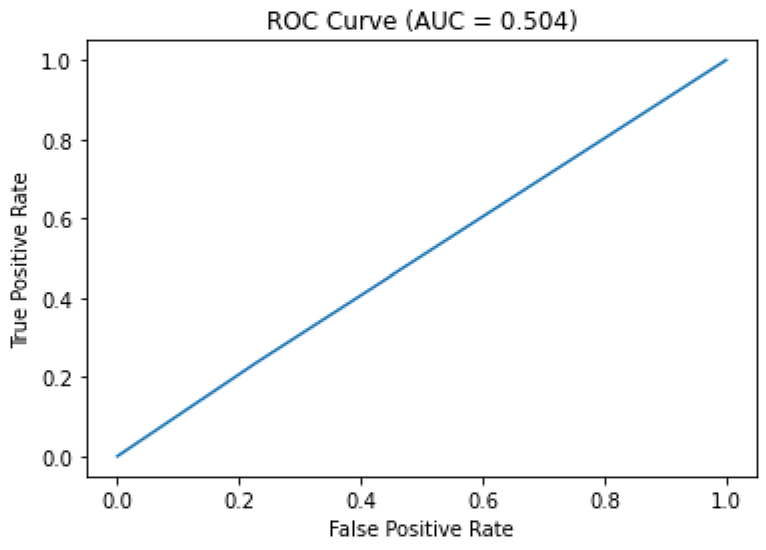
Gaussian SVM

```
Classification report:
              precision    recall  f1-score   support

         0.0       0.48      0.63      0.54       247
         1.0       0.46      0.31      0.37       247

    accuracy                           0.47       494
   macro avg       0.47      0.47      0.46       494
weighted avg       0.47      0.47      0.46       494
```



Cubic SVM

```
Classification report:
              precision    recall  f1-score   support

         0.0       0.50      0.77      0.61       247
         1.0       0.51      0.24      0.33       247

    accuracy                           0.50       494
   macro avg       0.51      0.50      0.47       494
weighted avg       0.51      0.50      0.47       494
```



FINDINGS & LEARNINGS

FINDINGS

Model	Accuracy	Precision	Recall	F1 Score	AUC - ROC
KNN (K=1)	0.51	0.53	0.56	0.44	0.564
KNN (K=3)	0.79	0.79	0.79	0.79	0.789
SVM (Linear)	0.71	0.80	0.71	0.69	0.709
Cubic SVM	0.50	0.51	0.50	0.47	0.504
Gaussian SVM	0.47	0.47	0.47	0.46	0.470
Naive Bayes	0.78	0.81	0.78	0.77	0.779
Logistic Regression	0.73	0.74	0.73	0.72	0.727
ANN	**0.87**	**0.87**	**0.87**	**0.87**	**0.874**

By comparing the classification reports and AUC-ROC of the models, it is found that ANN Model is having better scores than all the other models.

The ANN model has an accuracy of 87%.
Also the precision, Recall and F1 Score is also found to be 87%, The Area Under the ROC curve for ANN model is found to be 0.874, which is pretty good.

Naive Bayes is the next best model with accuracy of 78%, precision 81%, recall 78% and f1 score of 0.77.
The Area Under the ROC curve for Naive Bayes is found to be 0.779.

It is found that it is crucial to balance the dataset before fitting it to the model to avoid biased predictions.

LEARNINGS FROM THE PROJECT

- Understandings about how Artificial Neural Network models work.
- Learnings on various mathematical concepts associated with ANN
- Learnings about terms associated with ANN and their applications.
- Learnings on Tensorflow and Keras modules
- Understandings on various ML models and how they differ with each other.
- Learnings on various dataset balancing techniques
- Learnings on SMOTE and its benefits.
- Understandings on how to evaluate the classification report and plotting of ROC curve.

CONCLUSION

Employee Attrition is a critical issue for organizations as it can lead to a significant loss of talent, experience, and resources. Organizations need to understand the underlying reasons for employee attrition and take measures to retain their valuable employees. For that organizations can use various machine learning models to predict Employee Attrition with the help of historical data.

The artificial neural network (ANN) was used in this project to forecast employee attrition. The dataset was cleaned and preprocessed and balanced using SMOTE before fitting the data into the model for training.

The ANN model was built with Keras with TensorFlow as a backend, and it was optimized with the Adam optimizer. The model had 2 hidden layers with 100 nodes each with ‘relu’ activation function. The model predicted employee attrition with a high accuracy of 87%, demonstrating that the approach may be used to efficiently identify employees at danger of leaving the organization.
However using an Artificial dataset such as IBM HR Analytics Employee Attrition and Performance used in this study is not advised for better and realistic predictions.

Other traditional models such as Logistic Regression, Naive Bayes, KNN and SVM are also built and trained with the same balanced dataset. The comparison of classification reports and AUC-ROC showed that the scores of the ANN model were dominating over all the other models. After ANN, Naive Bayes is found to perform well compared to other models. K Nearest Neighbors with K=3 is also performing well in terms of classification.

Since the study shows that ANN model is effective among all the other models used in the study, it can be concluded that ANN is a best choice in predicting Employee Attritions of organizations.

REFERENCES

"Prediction of heart disease using ANNs" by A. Khademi et al. (2019)

Y. Wang et al, "ANN-based Stock Prediction System" (2019)

"Predicting COVID-19 Using Artificial Neural Networks" by Hassanien et al. (2021)

"ANN-Based Traffic Prediction for Smart Cities" by H. Lin et al. (2021)

https://www.geeksforgeeks.org/artificial-neural-networks-and-its-applications/

https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

https://www.v7labs.com/blog/neural-networks-activation-functions