DATA SCIENCE PROCESS PIPELINE FOR SOLVING EMPLOYEE ATTRITION AND THEIR JOB PERFORMANCE AND PREDICTING WITH AI

A Major Project Report submitted
in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology
in
Computer Science and Engineering

by

Vanka Hari Janardhan	**(16N31A05P9)**
S Vinay	**(16N31A05J6)**
Thota Manikanta Naga Hanuman	**(16N31A05N0)**

Under the esteemed guidance of

Mr. T Siva Ratna Sai
Assistant Professor

Department of Computer Science and Engineering
Malla Reddy College of Engineering & Technology
(Autonomous Institution- UGC, Govt. of India)
(Affiliated to JNTUH, Approved by AICTE, NBA & NAAC with 'A' Grade)
Maisammaguda, Kompally, Dhulapally, Secunderabad – 500100
website: www.mrcet.ac.in

2016-2020

CERTIFICATE

This is to certify that this is the bonafide record of the project entitled " Data Science Process Pipeline for solving Employee Attrition and their job performance and predicting with AI ", submitted by Vanka Hari Janardhan (16N31A05P9), S Vinay (16N31A05J6) and Thota Manikanta Naga Hanuman (16N31A05N0) of B.Tech in the partial fulfillment of the requirements for the degree of Bachelor of Technology in Computer Science and Engineering, Department of CSE during the year 2019-2020. The results embodied in this project report have not been submitted to any other university or institute for the award of any degree or diploma.

Internal Guide
Mr. T Siva Ratna Sai
Assistant Professor

Head of the Department
Dr. D. Sujatha
Professor

External Examiner

DECLARATION

We hereby declare that the project titled " Data Science Process Pipeline for solving Employee Attrition and their job performance and predicting with AI " submitted to Malla Reddy College of Engineering and Technology (UGC Autonomous), affiliated to Jawaharlal Nehru Technological University Hyderabad (JNTUH) for the award of the degree of Bachelor of Technology in Computer Science and Engineering is a result of original research carried-out in this thesis. It is further declared that the project report or any part thereof has not been previously submitted to any University or Institute for the award of degree or diploma.

Vanka Hari Janardhan (16N31A05P9)
S Vinay (16N31A05J6)
Thota Manikanta Naga Hanuman (16N31A05N0)

ACKNOWLEDGEMENT

We feel ourselves honored and privileged to place our warm salutation to our college Malla Reddy College of Engineering and Technology (UGC-Autonomous) and our principal Dr. VSK Reddy who gave us the opportunity to have experience in engineering and profound technical knowledge.

We express our heartiest thanks to our Head of the Department Dr. D. Sujatha for encouraging us in every aspect of our project and helping us realize our full potential.

We would like to thank our internal guide Mr. T Siva Ratna Sai and Project Coordinator Dr. S. Shanthi for their regular guidance and constant encouragement. We are extremely grateful to their valuable suggestions and unflinching co-operation throughout project work. We would like to thank our class in charge Mrs. V. Suneetha who in spite of being busy with her duties took time to guide and keep us on the correct path.

We would also like to thank all the supporting staff of the Department of CSE and all other departments who have been helpful directly or indirectly in making our project a success.

We are extremely grateful to our parents for their blessings and prayers for the completion of our project that gave us strength to do our project.

With regards and gratitude

Vanka Hari Janardhan (16N31A05P9)

S Vinay (16N31A05J6)

Thota Manikanta Naga Hanuman (16N31A05N0)

ABSTRACT

Among all employee related problems, employee attrition is one of the key problems in the today's scenario despite the changes in the external environment. Attrition is said to be gradual reduction in number of employees through resignation, death and retirement. When a well-trained and well-adapted employee leaves the organization for any of the reason, it creates an empty space in an organization (i.e.) there occurs a vacuum in the organization. It creates a great difficulty for a Human resource personnel to fill the gap that has occurred. Modern Human resource managers is taking various steps to reduce the employee attrition rate and it has been a pivotal challenge for today's Managers. Many of the employees may also tend to leave the job for various undisclosed factors such as lack of job security, lack of career advancement, desire for change in new opportunities, anticipating higher pay, problems with supervisors and few other personal reasons. This study helps in predicting employee attrition.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF OUTPUTS

(1) INTRODUCTION

Machine learning is a subfield of artificial intelligence (AI). The goal of machine learning generally is to understand the structure of data and fit that data into models that can be understood and utilized by people.

Although machine learning is a field within computer science, it differs from traditional computational approaches. In traditional computing, algorithms are sets of explicitly programmed instructions used by computers to calculate or problem solving. Machine learning algorithms instead allow for computers to train on data inputs and use statistical analysis in order to output values that fall within a specific range. Because of this, machine learning facilitates computers in building models from sample data in order to automate decision-making processes based on data inputs.

All technology users today have benefitted from machine learning. Facial recognition technology allows social media platforms to help users tag and share photos of friends. Optical character recognition (OCR) technology converts images of text into movable type. Recommendation engines, powered by machine learning, suggest what movies or television shows to watch next based on user preferences. Self-driving cars that rely on machine learning to navigate may soon be available to consumers.

Machine learning is a continuously developing field. Because of this, there are some considerations to keep in mind as you work with machine learning methodologies, or analyze the impact of machine learning processes.

Machine Learning vs. Traditional Programming

Traditional programming differs significantly from machine learning. In traditional programming, a programmer code all the rules in consultation with an expert in the industry for which software is being developed. Each rule is based on a logical foundation; the machine will execute an output following the logical statement. When the system grows complex, more rules need to be written. It can quickly become unsustainable to maintain.

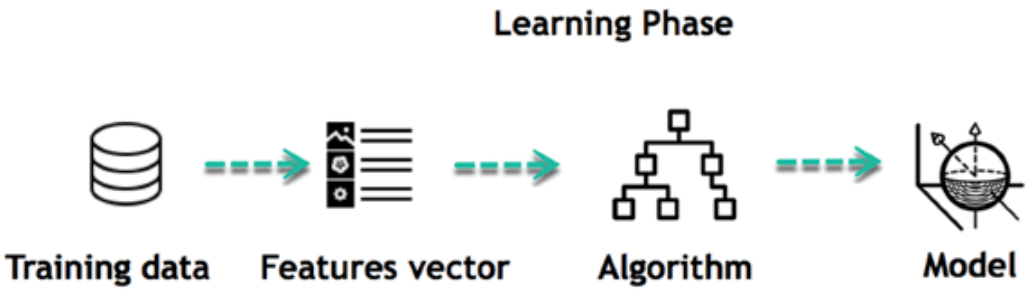


Figure 1.1: Learning Phase

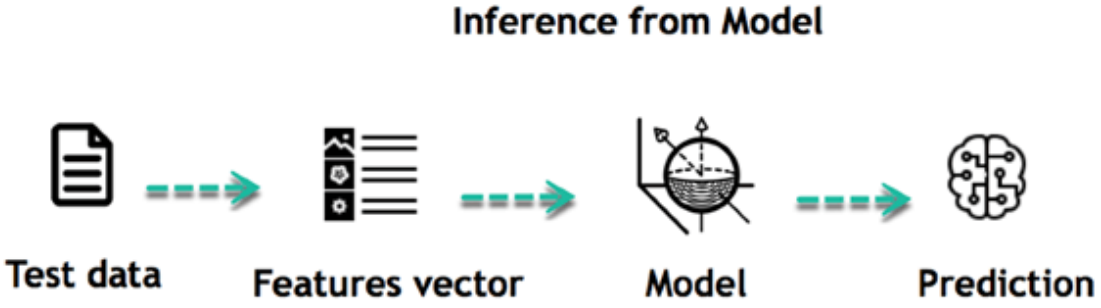


Figure 1.2: Inference from Model

(1.1) Machine Learning Methods

In machine learning, tasks are generally classified into broad categories. These categories are based on how learning is received or how feedback on the learning is given to the system developed.

Two of the most widely adopted machine learning methods are **supervised learning** which trains algorithms based on example input and output data that is labelled by humans, and **unsupervised learning** which provides the algorithm with no labelled data in order to allow it to find structure within its input data. Let's explore these methods in more detail.

(1.1.1) Supervised Learning

In supervised learning, the computer is provided with example inputs that are labelled with their desired outputs. The purpose of this method is for the algorithm to be able to "learn" by comparing its actual output with the "taught" outputs to find errors, and modify the model

accordingly. Supervised learning therefore uses patterns to predict label values on additional unlabelled data.

For example, with supervised learning, an algorithm may be fed data with images of sharks labelled as fish and images of oceans labelled as water. By being trained on this data, the supervised learning algorithm should be able to later identify unlabelled shark images as fish and unlabelled ocean images as water.

A common use case of supervised learning is to use historical data to predict statistically likely future events. It may use historical stock market information to anticipate upcoming fluctuations, or be employed to filter out spam emails. In supervised learning, tagged photos of dogs can be used as input data to classify untagged photos of dogs.

Algorithm Name	Description	Type
Linear regression	Finds a way to correlate each feature to the output to help predict future values.	Regression
Logistic regression	Extension of linear regression that's used for classification tasks. The output variable 3is binary (e.g., only black or white) rather than continuous (e.g., an infinite list of potential colors)	Classification
Decision tree	Highly interpretable classification or regression model that splits data-feature values into branches at decision nodes (e.g., if a feature is a color, each possible color becomes a new branch) until a final decision output is made	Regression Classification
Naive Bayes	The Bayesian method is a classification method that makes use of the Bayesian theorem. The theorem updates the prior knowledge of an event with the independent probability of each feature that can affect the event.	Regression Classification
Support vector machine	Support Vector Machine, or SVM, is typically used for the classification task. SVM algorithm finds a hyperplane that optimally divided the classes. It is best used with a non-linear solver.	Regression (not very common) Classification
Random forest	The algorithm is built upon a decision tree to improve the accuracy drastically. Random forest generates many times simple decision trees and uses the 'majority vote' method to decide on which label to return. For the classification task, the final prediction will be the one with the most vote; while for the regression task, the average prediction of all the trees is the final prediction.	Regression Classification

Table 1.1: Supervised Learning

(1.1.2) Unsupervised Learning

In unsupervised learning, data is unlabelled, so the learning algorithm is left to find commonalities among its input data. As unlabelled data are more abundant than labelled data, machine learning methods that facilitate unsupervised learning are particularly valuable. The goal of unsupervised learning may be as straightforward as discovering hidden patterns within a dataset, but it may also have a goal of feature learning, which allows the computational machine to automatically discover the representations that are needed to classify raw data.

Unsupervised learning is commonly used for transactional data. You may have a large dataset of customers and their purchases, but as a human you will likely not be able to make sense of what similar attributes can be drawn from customer profiles and their types of purchases. With this data fed into an unsupervised learning algorithm, it may be determined that women of a certain age range who buy unscented soaps are likely to be pregnant, and therefore a marketing campaign related to pregnancy and baby products can be targeted to this audience in order to increase their number of purchases. Unsupervised learning is often used for anomaly detection including for fraudulent credit card purchases, and recommender systems that recommend what products to buy next. In unsupervised learning, untagged photos of dogs can be used as input data for the algorithm to find likenesses and classify dog photos together.

Algorithm	Description	Type
K-means clustering	Puts data into some groups (k) that each contains data with similar characteristics (as determined by the model, not in advance by humans)	Clustering
Gaussian mixture model	A generalization of k-means clustering that provides more flexibility in the size and shape of groups (clusters	Clustering
Hierarchical clustering	Splits clusters along a hierarchical tree to form a classification system. Can be used for Cluster loyalty-card customer	Clustering
Recommender system	Help to define the relevant data for making a recommendation.	Clustering
PCA/T-SNE	Mostly used to decrease the dimensionality of the data. The algorithms reduce the number of features to 3 or 4 vectors with the highest variances.	Dimension Reduction

Table 1.2: Unsupervised Learning

(1.2) Concepts

DATA COLLECTION: Data used in this project is a set of employee details in the IBM records. This step is concerned with selecting the subset of all available data that you will be working with. ML problems start with data preferably, lots of data (examples or observations) for which you already know the target answer. Data for which you already know the target answer is called labelled data.

DATA PRE-PROCESSING: Organize your selected data by formatting, cleaning and sampling from it. Three common data pre-processing steps are: Formatting, Cleaning and Sampling.

FORMATTING: The data you have selected may not be in a format that is suitable for you to work with. The data may be in a relational database and you would like it in a flat file, or the data may be in a proprietary file format and you would like it in a relational database or a text file.

CLEANING: Cleaning data is the removal or fixing of missing data. There may be data instances that are incomplete and do not carry the data you believe you need to address the problem. These instances may need to be removed. Additionally, there may be sensitive information in some of the attributes and these attributes may need to be anonymized or removed from the data entirely.

SAMPLING: There may be far more selected data available than you need to work with. More data can result in much longer running times for algorithms and larger computational and memory requirements. You can take a smaller representative sample of the selected data that may be much faster for exploring and prototyping solutions before considering the whole dataset.

(1.3) Algorithms used in this Project

(1.3.1) Random Forest Classification

Random forest is a supervised learning algorithm which is used for both classification as well as regression. But however, it is mainly used for classification problems. As we know that a

forest is made up of trees and more trees means more robust forest. Similarly, random forest algorithm creates decision trees on data samples and then gets the prediction from each of them and finally selects the best solution by means of voting. It is an ensemble method which is better than a single decision tree because it reduces the over-fitting by averaging the result.

We can understand the working of Random Forest algorithm with the help of following steps

- **Step 1** − First, start with the selection of random samples from a given dataset.
- **Step 2** − Next, this algorithm will construct a decision tree for every sample. Then it will get the prediction result from every decision tree.
- **Step 3** − In this step, voting will be performed for every predicted result.
- **Step 4** − At last, select the most voted prediction result as the final prediction result.

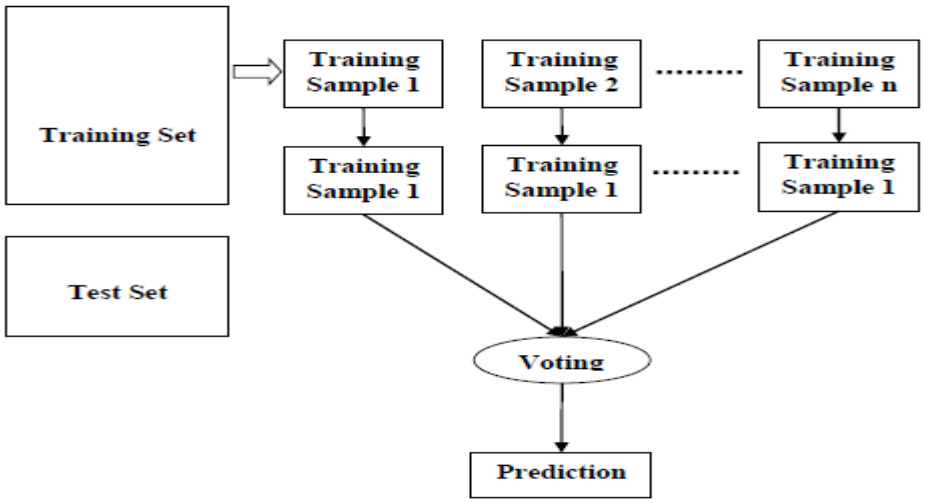


Figure 1.3: Random Forest Algorithm Architecture

ADVANTAGES OF USING RANDOM FOREST

1. The random forest algorithm is not biased, since there are multiple trees and each tree is trained on a subset of data. Basically, the random forest algorithm relies on the power of "the crowd" therefore, the overall biasedness of the algorithm is reduced.
2. This algorithm is very stable. Even if a new data point is introduced in the dataset the overall algorithm is not affected much since new data may impact one tree, but it is very hard for it to impact all the trees.

3. The random forest algorithm works well when you have both categorical and numerical features.
4. The random forest algorithm also works well when data has missing values or it has not been scaled.

(1.3.2) Logistic Regression

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of target or dependent variable is dichotomous, which means there would be only two possible classes.

In simple words, the dependent variable is binary in nature having data coded as either 1 (stands for success/yes) or 0 (stands for failure/no).

Mathematically, a logistic regression model predicts P(Y=1) as a function of X. It is one of the simplest ML algorithms that can be used for various classification problems such as spam detection, Diabetes prediction, cancer detection etc.

Generally, logistic regression means binary logistic regression having binary target variables, but there can be two more categories of target variables that can be predicted by it. Based on those number of categories, Logistic regression can be divided into following types

Binary or Binomial

In such a kind of classification, a dependent variable will have only two possible types either 1 and 0. For example, these variables may represent success or failure, yes or no, win or loss etc.

Multinomial

In such a kind of classification, dependent variable can have 3 or more possible **unordered** types or the types having no quantitative significance. For example, these variables may represent “Type A” or “Type B” or “Type C”.

Ordinal

In such a kind of classification, dependent variable can have 3 or more possible **ordered** types or the types having a quantitative significance. For example, these variables may represent "poor" or "good", "very good", "Excellent" and each category can have the scores like 0,1,2,3.

(1.3.3) Support Vector Machine

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. But generally, they are used in classification problems. In 1960s, SVMs were first introduced but later they got refined in 1990. SVMs have their unique way of implementation as compared to other machine learning algorithms. Lately, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

An SVM model is basically a representation of different classes in a hyperplane in multidimensional space. The hyperplane will be generated in an iterative manner by SVM so that the error can be minimized. The goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH).

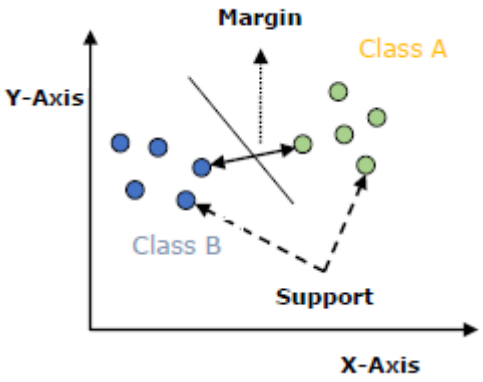


Figure 1.4: Support Vector Machine

The followings are important concepts in SVM

- **Support Vectors** − Datapoints that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.

- **Hyperplane** − As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.
- **Margin** − It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

The main goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH) and it can be done in the following two steps −

- First, SVM will generate hyperplanes iteratively that segregates the classes in best way.
- Then, it will choose the hyperplane that separates the classes correctly.

(1.3.4) Multi-Layer Perceptron Classifier

Multi-Layer perceptron defines the most complicated architecture of artificial neural networks. It is substantially formed from multiple layers of perceptron.

The diagrammatic representation of multi-layer perceptron learning is as shown below

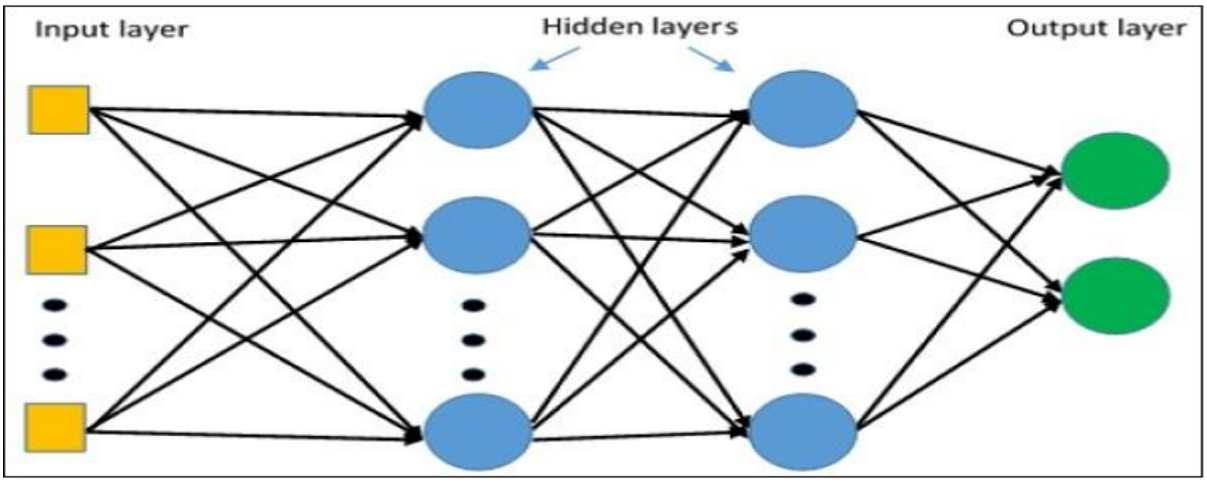


Figure 1.5: Multi-Layer Perceptron

MLP networks are usually used for supervised learning format. A typical learning algorithm for MLP networks is also called back propagation's algorithm.

(1.3.5) Data Science Process Pipeline

Data science pipelines are sequences of processing and analysis steps applied to data for a specific purpose. They're useful in production projects, and they can also be useful if one expects to encounter the same type of business question in the future, so as to save on design time and coding. For instance, one could remove outliers, apply dimensionality reduction techniques, and then run the result through a random forest classifier to provide automatic classification on a particular dataset that is pulled every week.

(1.4) Applications

Examples of machine learning applications

Augmentation:

- Machine learning, which assists humans with their day-to-day tasks, personally or commercially without having complete control of the output. Such machine learning is used in different ways such as Virtual Assistant, Data analysis, software solutions. The primary user is to reduce errors due to human bias.

Automation:

- Machine learning, which works entirely autonomously in any field without the need for any human intervention. For example, robots performing the essential process steps in manufacturing plants.

Finance Industry

- Machine learning is growing in popularity in the finance industry. Banks are mainly using ML to find patterns inside the data but also to prevent fraud.

Government organization

- The government makes use of ML to manage public safety and utilities. Take the example of China with the massive face recognition. The government uses Artificial intelligence to prevent jaywalker.

Healthcare industry

- Healthcare was one of the first industry to use machine learning with image detection.

Marketing

- Broad use of AI is done in marketing thanks to abundant access to data. Before the age of mass data, researchers develop advanced mathematical tools like Bayesian analysis to estimate the value of a customer. With the boom of data, marketing department relies on AI to optimize the customer relationship and marketing campaign.

Example of application of Machine Learning in Supply Chain

Machine learning gives terrific results for visual pattern recognition, opening up many potential applications in physical inspection and maintenance across the entire supply chain network. Unsupervised learning can quickly search for comparable patterns in the diverse dataset. In turn, the machine can perform quality inspection throughout the logistics hub, shipment with damage and wear.

For instance, IBM's Watson platform can determine shipping container damage. Watson combines visual and systems-based data to track, report and make recommendations in real-time. In past year stock manager relies extensively on the primary method to evaluate and forecast the inventory. When combining big data and machine learning, better forecasting techniques have been implemented (an improvement of 20 to 30 % over traditional forecasting tools). In term of sales, it means an increase of 2 to 3 % due to the potential reduction in inventory costs.

Example of Machine Learning Google Car

For example, everybody knows the Google car. The car is full of lasers on the roof which are telling it where it is regarding the surrounding area. It has radar in the front, which is informing the car of the speed and motion of all the cars around it. It uses all of that data to figure out not only how to drive the car but also to figure out and predict what potential drivers around the

car are going to do. What's impressive is that the car is processing almost a gigabyte a second of data.

Deep Learning

Deep learning is a computer software that mimics the network of neurons in a brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks. The machine uses different layers to learn from the data. The depth of the model is represented by the number of layers in the model. Deep learning is the new state of the art in term of AI. In deep learning, the learning phase is done through a neural network.

Reinforcement Learning

Reinforcement learning is a subfield of machine learning in which systems are trained by receiving virtual "rewards" or "punishments," essentially learning by trial and error. Google's DeepMind has used reinforcement learning to beat a human champion in the Go games. Reinforcement learning is also used in video games to improve the gaming experience by providing smarter bot.

One of the most famous algorithms are:

- Q-learning
- Deep Q network
- State-Action-Reward-State-Action (SARSA)
- Deep Deterministic Policy Gradient (DDPG)

Examples of deep learning applications

AI in Finance: The financial technology sector has already started using AI to save time, reduce costs, and add value. Deep learning is changing the lending industry by using more robust credit scoring. Credit decision-makers can use AI for robust credit lending applications to achieve faster, more accurate risk assessment, using machine intelligence to factor in the character and capacity of applicants.

Underwrite is a Fintech company providing an AI solution for credit makers company. underwrite.ai uses AI to detect which applicant is more likely to pay back a loan. Their approach radically outperforms traditional methods.

AI in HR: Under Armour, a sportswear company revolutionizes hiring and modernizes the candidate experience with the help of AI. In fact, Under Armour Reduces hiring time for its retail stores by 35%. Under Armour faced a growing popularity interest back in 2012. They had, on average, 30000 resumes a month. Reading all of those applications and begin to start the screening and interview process was taking too long. The lengthy process to get people hired and on-boarded impacted Under Armour's ability to have their retail stores fully staffed, ramped and ready to operate.

At that time, Under Armour had all of the 'must have' HR technology in place such as transactional solutions for sourcing, applying, tracking and onboarding but those tools weren't useful enough. Under armour choose HireVue, an AI provider for HR solution, for both on-demand and live interviews. The results were bluffing they managed to decrease by 35% the time to fill. In return, the hired higher quality staffs.

AI in Marketing: AI is a valuable tool for customer service management and personalization challenges. Improved speech recognition in call-center management and call routing as a result of the application of AI techniques allows a more seamless experience for customers. For example, deep-learning analysis of audio allows systems to assess a customer’s emotional tone. If the customer is responding poorly to the AI chatbot, the system can be rerouted the conversation to real, human operators that take over the issue.

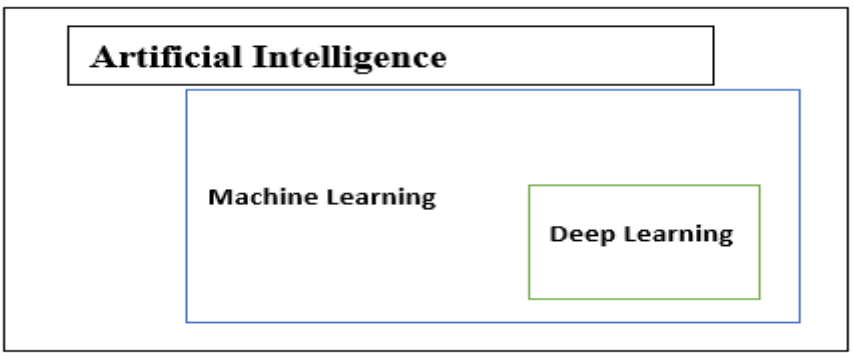


Figure 1.6: Categories of AI

Machine learning Algorithms and where they are used?

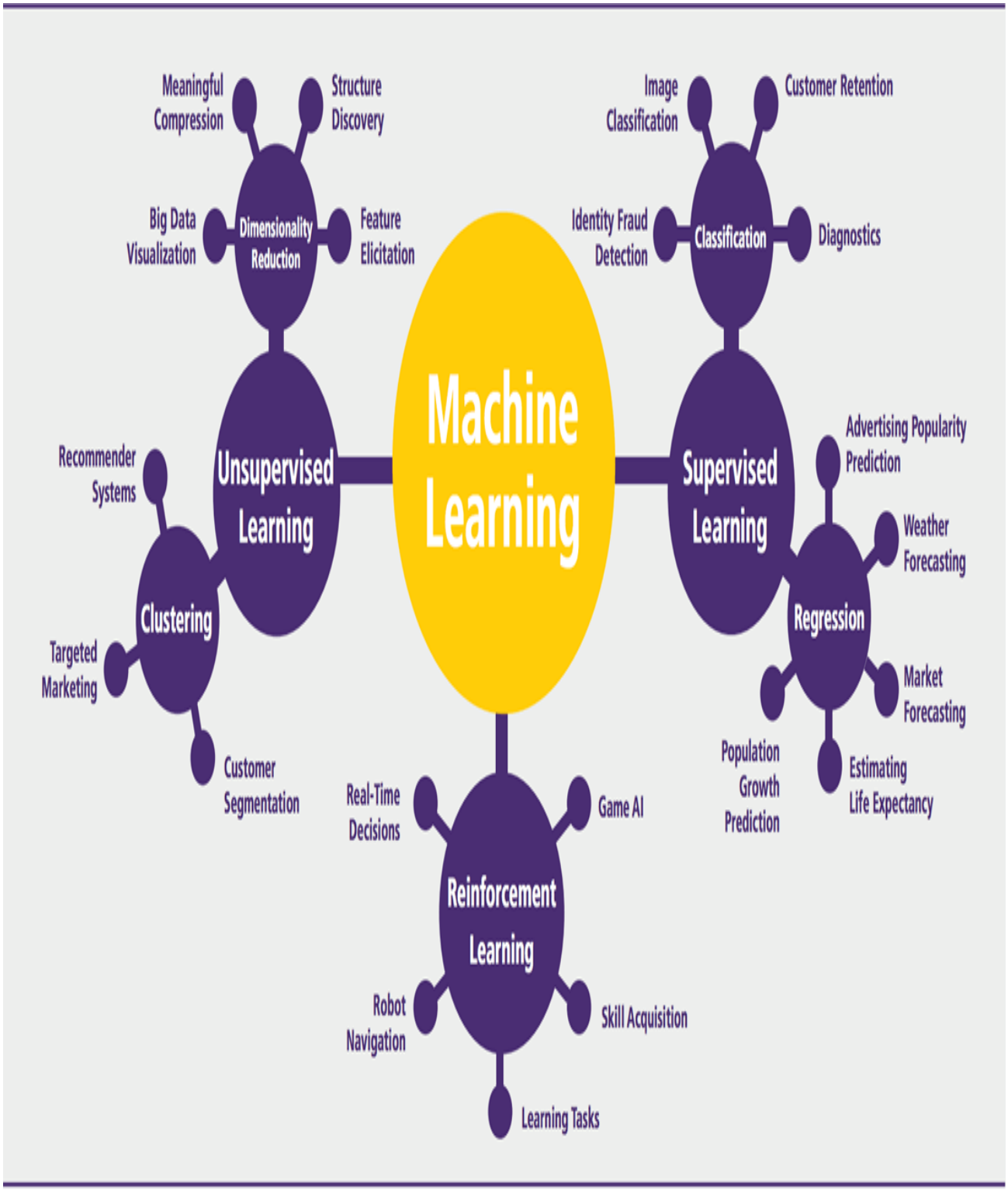


Figure 1.7: Applications of Machine Learning

(2) EXISTING SYSTEM

Now a day's data science predictions are used in IT industries, for the improvement in market investment, employee management etc. Retention of valuable employees within an organization has become an important issue as it is hard to find out the reasons that why employees are leaving an organization and keep them satisfied is a big challenge, for this a report is made to predict the retention of an employee in an organization using the python programming with data science methods. The main idea of this report is to find out that which valuable employee will leave the company and the features which are affecting him/her to making this decision like salary level, no. of hours spending in week, promotion, no. of work accident etc. The application was developed in python programming and are made with the help of data science and machine learning models. Earlier the results show on basis of information-sharing behaviors, information-seeking behaviors and general usage behaviors during working hours. Then shifted to some prediction analysis using Support vector machine, Naïve Bayes. The drawback of this system was it was time consuming as well as performance wise it is low, to increase the efficiency we moved to proposed methodology.

(3) PROPOSED SYSTEM

We know that larger companies contain more than thousand employees working for them, so taking care of the needs and satisfaction of each employee is a challenging task to do, it results in valuable and talented employees to leave the company without giving proper reasons. Employees are the assets of an organization. The main objective of the organization is to earn profit and to earn profit the employer should concentrate in retaining talents and concentrate in making them stick to the organization for the long run. Hence it is important for the employers to minimize the attrition rate and help in both individual as well as organizational growth. This project provides solution for the given problem as it gives a prediction model that can be used to predict which employee will leave the company and which will not leave. It also helps in finding the exact reasons which are motivating the employees for shifting companies like lower salary, less promotions or heavy work load etc. To find the result in the form of yes or no. Thus, organizations should create an environment that fosters ample growth opportunities, appreciation for the work accomplished and a friendly cooperative atmosphere that makes an employee feel connected in every respect to the organization.

(4) REQUIREMENTS

Hardware:

- RAM – 4GB

Software:

- OS – Windows 7, 8 and 10 (32 and 64 bit)
- Python IDLE
- Anaconda Navigator
- Python built-in modules
  - ✓ Numpy
  - ✓ Pandas
  - ✓ Matplotlib
  - ✓ Sklearn
  - ✓ Seaborn

A sample dataset looks like the following.

Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	EnvironmentSatisfaction	Gender	HourlyRate
41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	2	Female	94
49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	3	Male	61
37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	4	Male	92
33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	4	Female	56
27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	1	Male	40
32	No	Travel_Frequently	1005	Research & Development	2	2	Life Sciences	1	8	4	Male	79
59	No	Travel_Rarely	1324	Research & Development	3	3	Medical	1	10	3	Female	81
30	No	Travel_Rarely	1358	Research & Development	24	1	Life Sciences	1	11	4	Male	67
38	No	Travel_Frequently	216	Research & Development	23	3	Life Sciences	1	12	4	Male	44
36	No	Travel_Rarely	1299	Research & Development	27	3	Medical	1	13	3	Male	94
35	No	Travel_Rarely	809	Research & Development	16	3	Medical	1	14	1	Male	84
29	No	Travel_Rarely	153	Research & Development	15	2	Life Sciences	1	15	4	Female	49
31	No	Travel_Rarely	670	Research & Development	26	1	Life Sciences	1	16	1	Male	31
34	No	Travel_Rarely	1346	Research & Development	19	2	Medical	1	18	2	Male	93
28	Yes	Travel_Rarely	103	Research & Development	24	3	Life Sciences	1	19	3	Male	50
29	No	Travel_Rarely	1389	Research & Development	21	4	Life Sciences	1	20	2	Female	51
32	No	Travel_Rarely	334	Research & Development	5	2	Life Sciences	1	21	1	Male	80
22	No	Non-Travel	1123	Research & Development	16	2	Medical	1	22	4	Male	96
53	No	Travel_Rarely	1219	Sales	2	4	Life Sciences	1	23	1	Female	78
38	No	Travel_Rarely	371	Research & Development	2	3	Life Sciences	1	24	4	Male	45
24	No	Non-Travel	673	Research & Development	11	2	Other	1	26	1	Female	96
36	Yes	Travel_Rarely	1218	Sales	9	4	Life Sciences	1	27	3	Male	82

HR-Employee-Attrition

Table 4.1: Sample HR Attrition Dataset

(4.1) Anaconda Navigator

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows you to launch applications and easily manage conda packages, environments and channels without using command-line commands. Navigator can search for packages on Anaconda Cloud or in a local Anaconda Repository. It is available for Windows, mac OS and Linux.

Why use Navigator?

In order to run, many scientific packages depend on specific versions of other packages. Data scientists often use multiple versions of many packages, and use multiple environments to separate these different versions.

The command line program conda is both a package manager and an environment manager, to help data scientists ensure that each version of each package has all the dependencies it requires and works correctly.

Navigator is an easy, point-and-click way to work with packages and environments without needing to type conda commands in a terminal window. You can use it to find the packages you want, install them in an environment, run the packages and update them, all inside Navigator.

What applications can I access using the Navigator?

The following applications are available by default in Navigator:

- JupyterLab
- Jupyter Notebook
- QTConsole
- Spyder
- VSCode
- Glueviz
- Orange 3 App
- Rodeo
- RStudio

How can I run code with Navigator?

The simplest way is with Spyder. From the Navigator Home tab, click Spyder, and write and execute your code.

You can also use Jupyter Notebooks the same way. Jupyter Notebooks are an increasingly popular system that combine your code, descriptive text, output, images and interactive interfaces into a single notebook file that is edited, viewed and used in a web browser.

What's new in 1.9?

- Add support for **Offline Mode** for all environment related actions.
- Add support for custom configuration of main windows links.
- Numerous bug fixes and performance enhancements.

(4.2) Python Overview

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.

- **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

- **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.

- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, Unix shell, and other scripting languages. Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL). Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Python Features

Python's features include:

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.

- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.

- **Easy-to-maintain:** Python's source code is fairly easy-to-maintaining.

- **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.

- **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.

- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.

- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.

- **Databases:** Python provides interfaces to all major commercial databases.

- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries, and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

- **Scalable:** Python provides a better structure and support for large programs than shell scripting.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below:

- IT supports functional and structured programming methods as well as OOP.

- It can be used as a scripting language or can be compiled to byte-code for building large applications.

- It provides very high-level dynamic data types and supports dynamic type checking.

- IT supports automatic garbage collection.

- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.

Python's standard library

- Pandas
- Numpy
- Sklearn
- seaborn
- matplotlib
- Importing Datasets

(4.3) Pandas

Pandas is quite a game changer when it comes to analyzing data with Python and it is one of the most preferred and widely used tools in data munging/wrangling if not the most used one. Pandas is an open source tool.

What's cool about Pandas is that it takes data (like a CSV or TSV file, or a SQL database) and creates a Python object with rows and columns called data frame that looks very similar to table in a statistical software (think Excel or SPSS for example. People who are familiar with R would see similarities to R too). This is so much easier to work with in comparison to working with lists and/or dictionaries through for loops or list comprehension.

Installation and Getting Started

In order to "get" Pandas you would need to install it. You would also need to have Python 2.7 and above as a pre-requirement for installation. It is also dependent on other libraries (like NumPy) and has optional dependencies (like Matplotlib for plotting). Therefore, I think that the easiest way to get Pandas set up is to install it through a package like the Anaconda distribution , "a cross platform distribution for data analysis and scientific computing."

In order to use Pandas in your Python IDE (Integrated Development Environment) like Jupyter Notebook or Spyder (both of them come with Anaconda by default), you need to import the Pandas library first. Importing a library means loading it into the memory and then it's there for you to work with. In order to import Pandas all, you have to do is run the following code:

- **import pandas as pd**
- **import numpy as np**

Usually you would add the second part ('as pd') so you can access Pandas with 'pd. command' instead of needing to write 'pandas. command' every time you need to use it. Also, you would import numpy as well, because it is very useful library for scientific computing with Python.

Now Pandas is ready for use! Remember, you would need to do it every time you start a new Jupyter Notebook, Spyder file etc.

Working with Pandas

Loading and Saving Data with Pandas

When you want to use Pandas for data analysis, you'll usually use it in one of three different ways:

- Convert a Python's list, dictionary or Numpy array to a Pandas data frame
- Open a local file using Pandas, usually a CSV file, but could also be a delimited text file (like TSV), Excel, etc.
- Open a remote file or database like a CSV or a JSON on a website through a URL or read from a SQL table/database

There are different commands to each of these options, but when you open a file, they would look like this:

- **pd. read_filetype ()**

As mentioned before, there are different filetypes Pandas can work with, so you would replace "filetype" with the actual, well, filetype (like CSV). You would give the path, filename etc inside the parenthesis. Inside the parenthesis you can also pass different arguments that relate to how to open the file. There are numerous arguments and in order to know all you them, you would have to read the documentation (for example, the documentation for pd.read_csv() would contain all the arguments you can pass in this Pandas command).

In order to convert a certain Python object (dictionary, lists etc) the basic command is:

- **pd. DataFrame ()**

Inside the parenthesis you would specify the object(s) you're creating the data frame from. This command also has different arguments .

You can also save a data frame you're working with/on to different kinds of files (like CSV, Excel, JSON and SQL tables). The general code for that is:

- **df.to_filetype(filename)**

Viewing and Inspecting Data

Now that you've loaded your data, it's time to take a look. How does the data frame look? Running the name of the data frame would give you the entire table, but you can also get the first n rows with df. head(n) or the last n rows with df. tail(n). df. shape would give you the number of rows and columns. df.info () would give you the index, datatype and memory information. The command s.value_counts(dropna=False) would allow you to view unique values and counts for a series (like a column or a few columns). A very useful command is df. describe () which inputs summary statistics for numerical columns. It is also possible to get statistics on the entire data frame or a series (a column etc):

- df. mean () Returns the mean of all columns
- df. corr () Returns the correlation between columns in a data frame
- df. count () Returns the number of non-null values in each data frame column
- df.max () Returns the highest value in each column
- df.min () Returns the lowest value in each column
- df. median () Returns the median of each column
- df.std () Returns the standard deviation of each column

Selection of Data

One of the things that is so much easier in Pandas is selecting the data you want in comparison to selecting a value from a list or a dictionary. You can select a column (df[col]) and return column with label col as Series or a few columns (df [[col1, col2]]) and returns columns as a new DataFrame. You can select by position (s. iloc [0]), or by index (s.loc['index_one']). In order to select the first row, you can use df. iloc [0:] and in order to select the first element of

the first column you would run df. iloc [0,0]. These can also be used in different combinations, so I hope it gives you an idea of the different selection and indexing you can perform in Pandas.

Filter, Sort and Group by

You can use different conditions to filter columns. For example, df[df[year] > 1984] would give you only the column year is greater than 1984. You can use & (and) or | (or) to add different conditions to your filtering. This is also called Boolean filtering.

It is possible to sort values in a certain column in an ascending order using df. sort_values(col1); and also, in a descending order using df. sort_values (col2, ascending=False). Furthermore, it's possible to sort values by col1 in ascending order then col2 in descending order by using df. sort_values ([col1, col2], ascending= [True, False]).

The last command in this section is group by. It involves splitting the data into groups based on some criteria, applying a function to each group independently and combining the results into a data structure. df. groupby(col) returns a group by object for values from one column while df. groupby ([col1, col2]) returns a group by object for values from multiple columns.

Data Cleaning

Data cleaning is a very important step in data analysis. For example, we always check for missing values in the data by running pd.is null () which checks for null Values, and returns a Boolean array (an array of true for missing values and false for non-missing values). In order to get a sum of null/missing values, run pd. Is null (). sum (). Pd. not null () is the opposite of pd. Is null (). After you get a list of missing values you can get rid of them, or drop them by using df. Drop na () to drop the rows or df. drop na(axis=1) to drop the columns. A different approach would be to fill the missing values with other values by using df. Fill na(x) which fills the missing values with x (you can put there whatever you want) or s. fill na (s. mean ()) to replace all null values with the mean (mean can be replaced with almost any function from the statistics section).

It is sometimes necessary to replace values with different values. For example, s. replaces(1,'one') would replace all values equal to 1 with 'one'. It's possible to do it for multiple values: s. replace ([1,3], ['one', 'three']) would replace all 1 with 'one' and 3 with 'three'. You can also rename specific columns by running: df. rename (columns= {'old_name': 'new_ name'}) or use df. set_ index('column_one') to change the index of the data frame.

Join/Combine

The last set of basic Pandas commands are for joining or combining data frames or rows/columns. The three commands are:

- df1.append(df2)— add the rows in df1 to the end of df2 (columns should be identical)
- df. concat ([df1, df2], axis=1)—add the columns in df1 to the end of df2 (rows should be identical)
- df1.join(df2, on=col1, how='inner')—SQL-style join the columns in df1with the columns on df2 where the rows for col have identical values. how can be equal to one of: 'left', 'right', 'outer', 'inner'

(4.4) Numpy

Numpy is one such powerful library for array processing along with a large collection of high-level mathematical functions to operate on these arrays. These functions fall into categories like Linear Algebra, Trigonometry, Statistics, Matrix manipulation, etc.

NumPy's main object is a homogeneous multidimensional array. Unlike python's array class which only handles one-dimensional array, NumPy's nd array class can handle multidimensional array and provides more functionality. NumPy's dimensions are known as axes. For example, the array below has 2 dimensions or 2 axes namely rows and columns. Sometimes dimension is also known as a rank of that particular array or matrix.

Importing NumPy

NumPy is imported using the following command. Note here np is the convention followed for the alias so that we don't need to write numpy every time.

- import numpy as np

NumPy is the basic library for scientific computations in Python and this article illustrates some of its most frequently used functions. Understanding NumPy is the first major step in the journey of machine learning and deep learning.

(4.5) Sci-Kit Learn

In python, scikit-learn library has a pre-built functionality under sk learn. Preprocessing.
Next thing is to do feature extraction Feature extraction is an attribute reduction process. Unlike feature selection, which ranks the existing attributes according to their predictive significance, feature extraction actually transforms the attributes. The transformed attributes, or features, are linear combinations of the original attributes. Finally, our models are trained using Classifier algorithm. We use nltk. classify module on Natural Language Toolkit library on Python. We use the labelled dataset gathered. The rest of our labelled data will be used to evaluate the models. Some machine learning algorithms were used to classify preprocessed data. The chosen classifiers were Decision tree, Support Vector Machines and Random forest. These algorithms are very popular in text classification tasks.

Data Visualization in Python

Data visualization is the discipline of trying to understand data by placing it in a visual context, so that patterns, trends and correlations that might not otherwise be detected can be exposed. Python offers multiple great graphing libraries that come packed with lots of different features. No matter if you want to create interactive, live or highly customized plots python has an excellent library for you.

To get a little overview here are a few popular plotting libraries:

- **Matplotlib:** low level, provides lots of freedom
- **Pandas Visualization:** easy to use interface, built on Matplotlib
- **Seaborn:** high-level interface, great default styles
- **ggplot:** based on R's ggplot2, uses *Grammar of Graphics*
- **Plotly:** can create interactive plots

In this documentation, we will learn how to create basic plots using Matplotlib, Pandas visualization and Seaborn as well as how to use some specific features of each library.

(4.6) Matplotlib

Matplotlib is the most popular python plotting library. It is a low-level library with a MATLAB like interface which offers lots of freedom at the cost of having to write more code.

1. To install Matplotlib pip anaconda can be used.

2. pip install matplotlib

3. conda install matplotlib

Matplotlib is specifically good for creating basic graphs like line charts, bar charts, histograms and many more. It can be imported by typing:

- import matplotlib. pyplot as plt

Line Chart

In Matplotlib we can create a line chart by calling the plot method. We can also plot multiple columns in one graph, by looping through the columns we want, and plotting each column on the same axis.

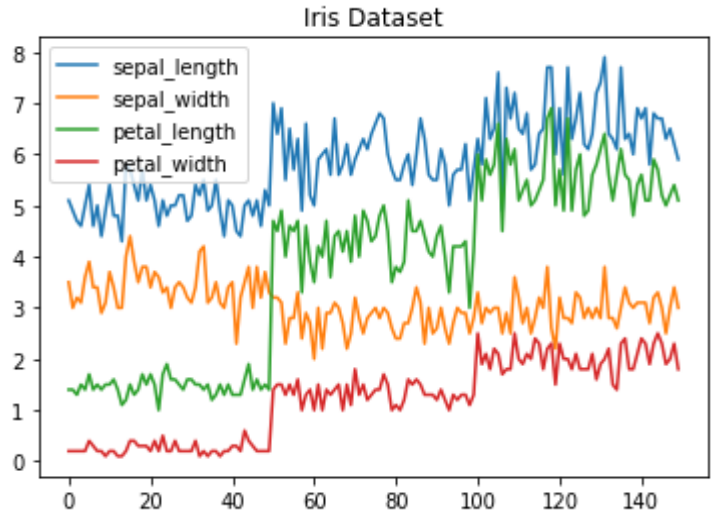


Figure 4.1: Line Chart

Histogram

In Matplotlib we can create a Histogram using the hist method. If we pass it categorical data like the points column from the wine-review dataset it will automatically calculate how often each class occurs.

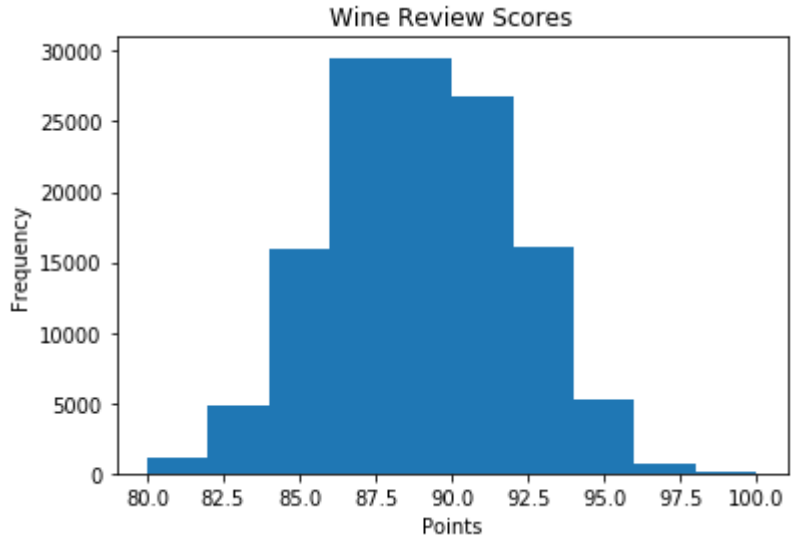


Figure 4.2: Histogram

Bar Chart

A bar-chart can be created using the bar method. The bar-chart isn't automatically calculating the frequency of a category so we are going to use pandas' value_counts function to do this. The bar-chart is useful for categorical data that doesn't have a lot of different categories (less than 30) because else it can get quite messy.

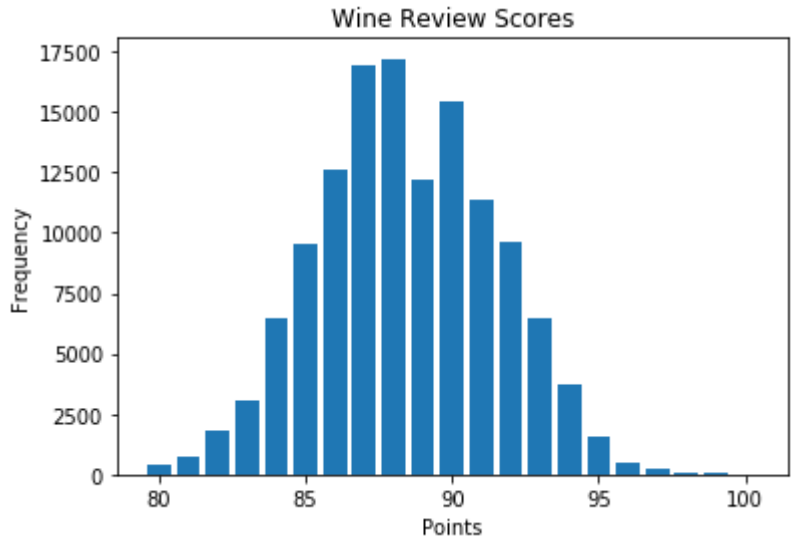


Figure 4.3: Bar-Chart

Heatmap

A Heatmap is a graphical representation of data where the individual values contained in a matrix are represented as colors. Heatmaps are perfect for exploring the correlation of features in a dataset.

To get the correlation of the features inside a dataset we can call <dataset>. corr (), which is a Pandas DataFrame method. This will give use the correlation matrix.

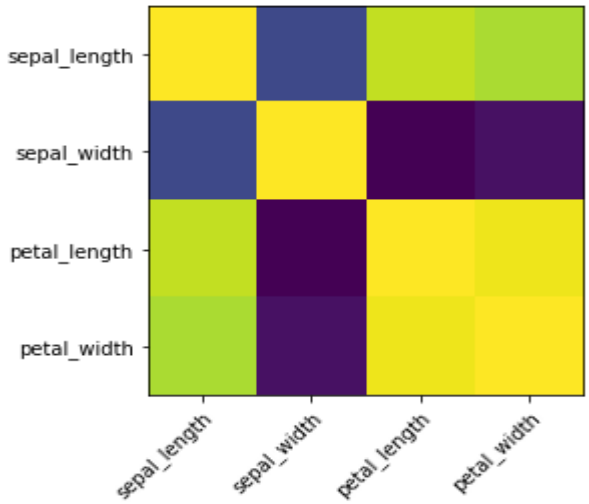


Figure 4.4: Heatmap without annotation

(5) ARCHITECTURES

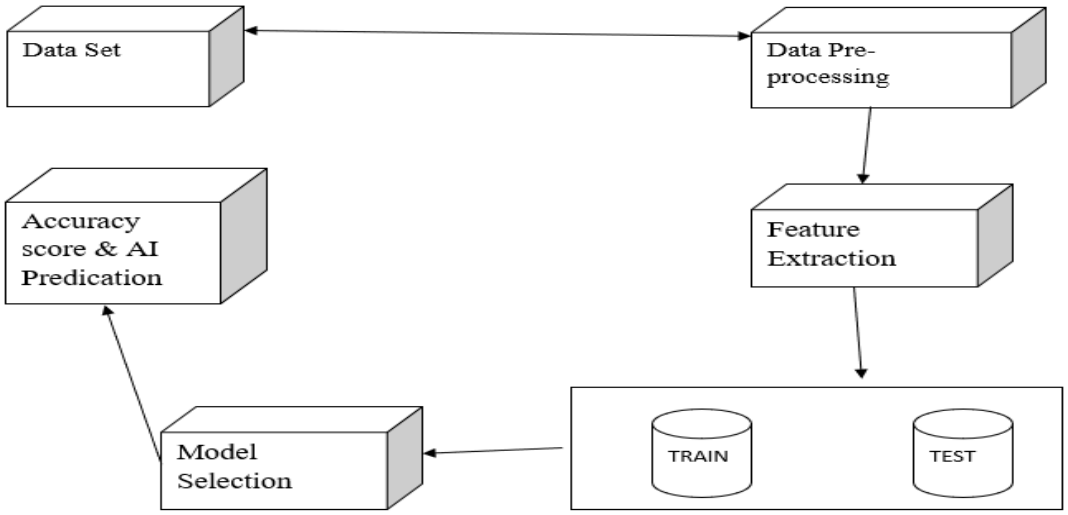


Figure 5.1: System Architecture

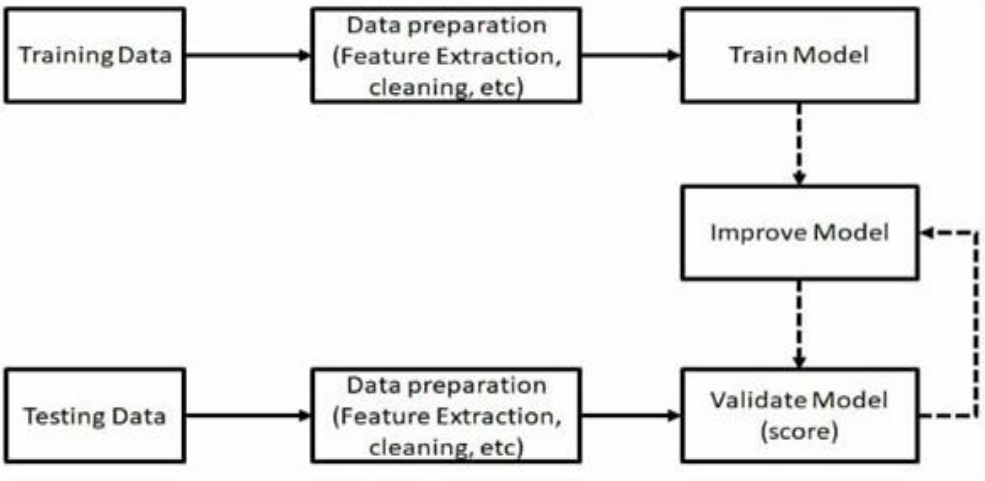


Figure 5.2: Technical Architecture

(6) STEPS PERFORMED IN THE PROJECT

1. Download Employee Dataset from IBM Website
2. Read the dataset in Jupyter Notebook in CSV format.
3. The python program must perform data pre-processing. We must identify the most important attributes of our dataset.
4. After choosing the most important attributes, we perform various machine learning algorithms.
5. We run the algorithm and the machine is trained and tested using the dataset in an 80-20 manner.
6. The algorithm which returns the best accuracy score in the confusion matrix is chosen as our algorithm in the project.
7. Take input values from end user using command prompt.
8. Based on the user input given in the command prompt, a prediction is made whether that particular employee leaves or stays using pipeline methods.
9. The result is displayed back on the command prompt.

(7) UML DIAGRAMS

The unified modeling is a standard language for specifying, visualizing, constructing and documenting the system and its components is a graphical language which provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure and control information about the systems.

Depending on the development culture, some of these artifacts are treated more or less formally than others. Such artifacts are not only the deliverables of a project; they are also critical in controlling, measuring, and communicating about a system during its development and after its deployment.

The UML addresses the documentation of a system's architecture and all of its details. The UML also provides a language for expressing requirements and for tests. Finally, the UML provides a language for modeling the activities of project planning and release management.

Building blocks of UML:

The vocabulary of the UML encompasses three kinds of building blocks:

- Things
- Relationships
- Diagrams

Things in UML:

There are four kinds of things in the UML:

- Structural things
- Behavioral things
- Grouping things
- Annotational things

(7.1) CLASS DIAGRAM

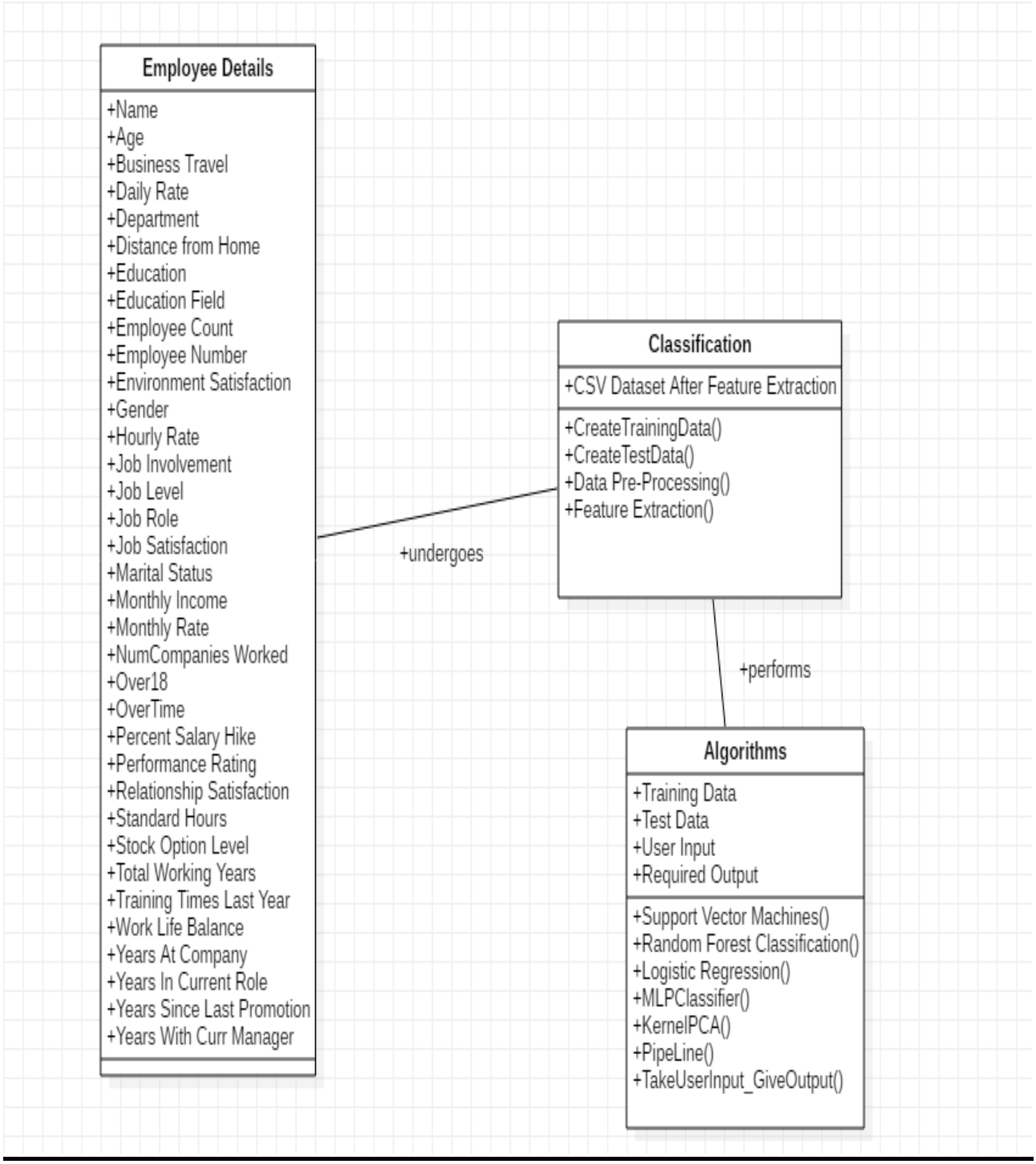


Figure 7.1: Class Diagram

(7.2) USE-CASE DIAGRAM

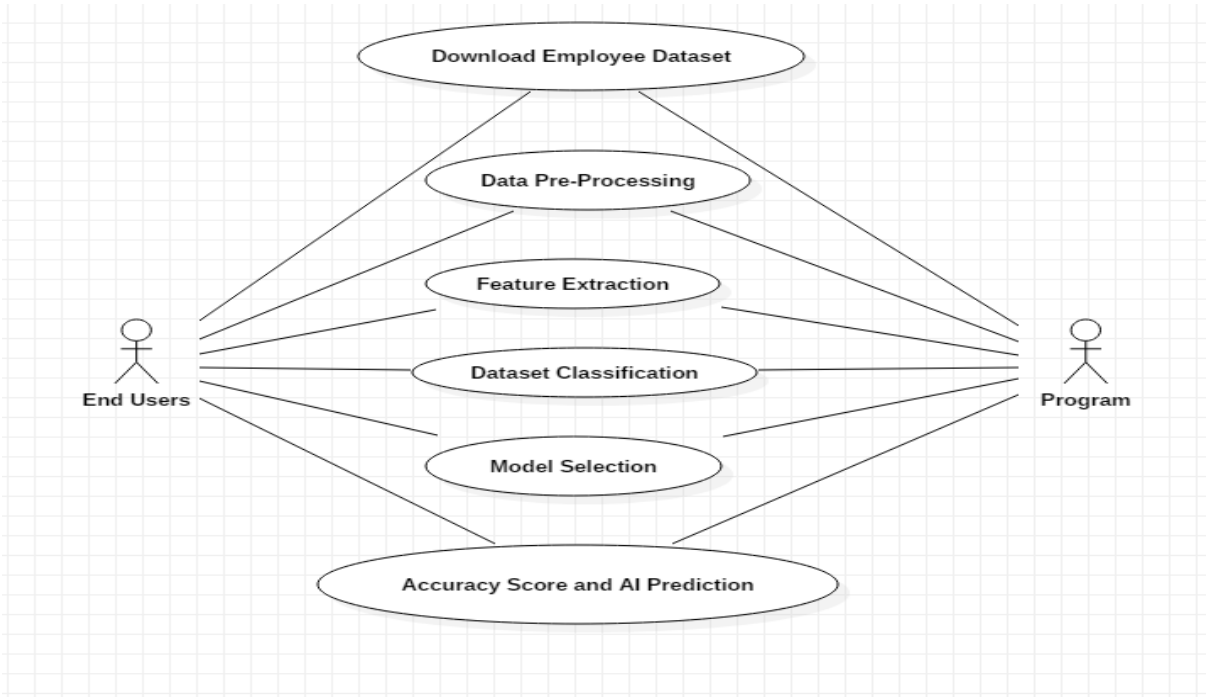


Figure 7.2: Use-Case Diagram

(7.3) ACTIVITY DIAGRAM

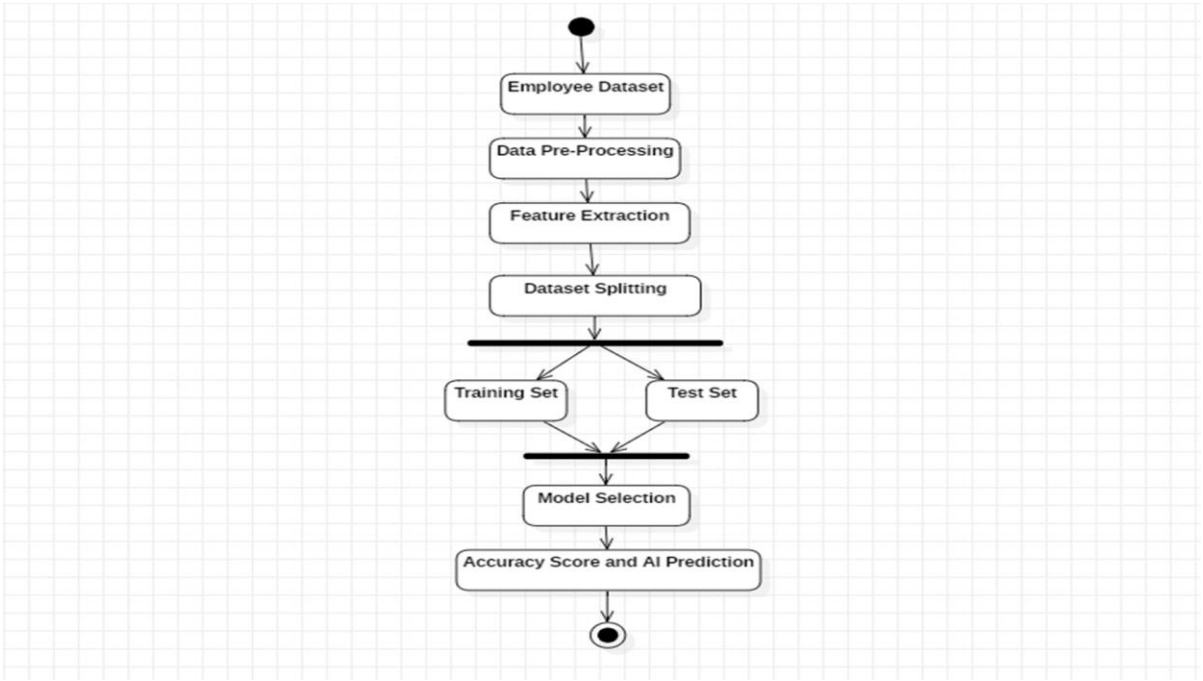


Figure 7.3: Activity Diagram

(7.4) SEQUENCE DIAGRAM

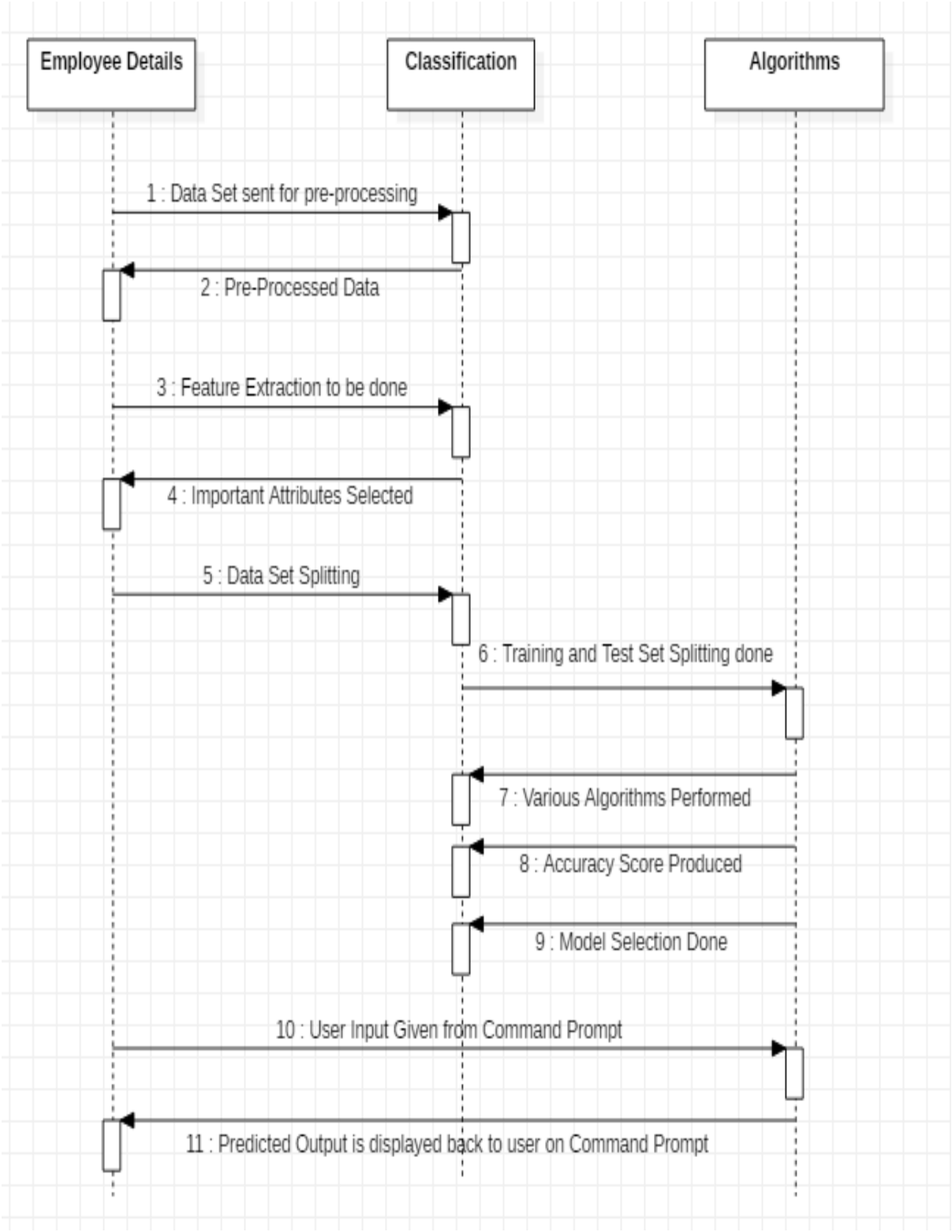


Figure 7.4: Sequence Diagram

(8) CODING PART WITH OUTPUTS

```
In [1]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt
        from sklearn import preprocessing
        #from sklearn.ensemble import AdaBoostClassifier
        from sklearn import metrics
        from sklearn.pipeline import Pipeline
        from sklearn.model_selection import train_test_split
        import warnings
        warnings.filterwarnings("ignore")
```

```
In [2]: df=pd.read_csv('EMPdataset.csv')
```

```
In [3]: df.head()
```

Out[3]:

	Age	Attrition	BusinessTravel	DailyRate	Department	DistanceFromHome	Education	EducationField	EmployeeCount	EmployeeNumber	...	Relationship
0	41	Yes	Travel_Rarely	1102	Sales	1	2	Life Sciences	1	1	...	
1	49	No	Travel_Frequently	279	Research & Development	8	1	Life Sciences	1	2	...	
2	37	Yes	Travel_Rarely	1373	Research & Development	2	2	Other	1	4	...	
3	33	No	Travel_Frequently	1392	Research & Development	3	4	Life Sciences	1	5	...	
4	27	No	Travel_Rarely	591	Research & Development	2	1	Medical	1	7	...	

```
In [4]: df.shape
```

```
Out[4]: (1470, 35)
```

```
In [5]: df.columns
```

```
Out[5]: 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')
```

Figure 8.1: Importing Modules and Dataset Pre-Processing

```
In [6]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1470 entries, 0 to 1469
Data columns (total 35 columns):
Age                         1470 non-null int64
Attrition                   1470 non-null object
BusinessTravel              1470 non-null object
DailyRate                   1470 non-null int64
Department                  1470 non-null object
DistanceFromHome            1470 non-null int64
Education                   1470 non-null int64
EducationField              1470 non-null object
EmployeeCount               1470 non-null int64
EmployeeNumber              1470 non-null int64
EnvironmentSatisfaction     1470 non-null int64
Gender                      1470 non-null object
HourlyRate                  1470 non-null int64
JobInvolvement              1470 non-null int64
JobLevel                    1470 non-null int64
JobRole                     1470 non-null object
JobSatisfaction             1470 non-null int64
MaritalStatus               1470 non-null object
MonthlyIncome               1470 non-null int64
MonthlyRate                 1470 non-null int64
NumCompaniesWorked          1470 non-null int64
Over18                      1470 non-null object
OverTime                    1470 non-null object
PercentSalaryHike           1470 non-null int64
PerformanceRating           1470 non-null int64
RelationshipSatisfaction    1470 non-null int64
StandardHours               1470 non-null int64
StockOptionLevel            1470 non-null int64
TotalWorkingYears           1470 non-null int64
TrainingTimesLastYear       1470 non-null int64
WorkLifeBalance             1470 non-null int64
YearsAtCompany              1470 non-null int64
YearsInCurrentRole          1470 non-null int64
YearsSinceLastPromotion     1470 non-null int64
YearsWithCurrManager        1470 non-null int64
dtypes: int64(26), object(9)
memory usage: 402.0+ KB
```

```
In [7]: data = df.drop(['DailyRate', 'EducationField', 'EmployeeCount', 'EmployeeNumber', 'HourlyRate', 'MonthlyRate','Over18', 'Relatio
```

```
In [8]: data.isnull().sum()
```

```
Out[8]: Age                         0
        Attrition                   0
        BusinessTravel              0
        Department                  0
        DistanceFromHome            0
        Education                   0
        EnvironmentSatisfaction     0
        Gender                      0
        JobInvolvement              0
        JobLevel                    0
        JobRole                     0
        JobSatisfaction             0
        MaritalStatus               0
        MonthlyIncome               0
        NumCompaniesWorked          0
        OverTime                    0
        PercentSalaryHike           0
        PerformanceRating           0
        StockOptionLevel            0
        TotalWorkingYears           0
        TrainingTimesLastYear       0
        WorkLifeBalance             0
        YearsAtCompany              0
        YearsInCurrentRole          0
        YearsSinceLastPromotion     0
        YearsWithCurrManager        0
        dtype: int64
```

Figure 8.2: Feature Extraction and Null Values Check

```
In [9]: data.head()
```

Out[9]:

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EnvironmentSatisfaction	Gender	JobInvolvement	JobLevel	...	PercentSalaryI
0	41	Yes	Travel_Rarely	Sales	1	2	2	Female	3	2	...	
1	49	No	Travel_Frequently	Research & Development	8	1	3	Male	2	2	...	
2	37	Yes	Travel_Rarely	Research & Development	2	2	4	Male	2	1	...	
3	33	No	Travel_Frequently	Research & Development	3	4	4	Female	3	1	...	
4	27	No	Travel_Rarely	Research & Development	2	1	1	Male	3	1	...	

5 rows × 26 columns

```
In [10]: data.shape
```

Out[10]: (1470, 26)

```
In [11]: categorical_column = ['Attrition', 'BusinessTravel', 'Department',
                              'Gender', 'JobRole', 'MaritalStatus', 'OverTime']
```

```
In [12]: data_encoded = data.copy(deep=True)
```

```
In [13]: lab_enc = preprocessing.LabelEncoder()
         for col in categorical_column:
                 data_encoded[col] = lab_enc.fit_transform(data[col])
                 le_name_mapping = dict(zip(lab_enc.classes_, lab_enc.transform(lab_enc.classes_)))
                 print('Feature', col)
                 print('mapping', le_name_mapping)
```

```
Feature Attrition
mapping {'No': 0, 'Yes': 1}
Feature BusinessTravel
mapping {'Non-Travel': 0, 'Travel_Frequently': 1, 'Travel_Rarely': 2}
Feature Department
mapping {'Human Resources': 0, 'Research & Development': 1, 'Sales': 2}
Feature Gender
mapping {'Female': 0, 'Male': 1}
Feature JobRole
mapping {'Healthcare Representative': 0, 'Human Resources': 1, 'Laboratory Technician': 2, 'Manager': 3, 'Manufacturing Directo
r': 4, 'Research Director': 5, 'Research Scientist': 6, 'Sales Executive': 7, 'Sales Representative': 8}
Feature MaritalStatus
mapping {'Divorced': 0, 'Married': 1, 'Single': 2}
Feature OverTime
mapping {'No': 0, 'Yes': 1}
```

```
In [14]: data_encoded.head()
```

Out[14]:

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EnvironmentSatisfaction	Gender	JobInvolvement	JobLevel	...	PercentSalaryHil
0	41	1	2	2	1	2	2	0	3	2	...	
1	49	0	1	1	8	1	3	1	2	2	...	
2	37	1	2	1	2	2	4	1	2	1	...	
3	33	0	1	1	3	4	4	0	3	1	...	
4	27	0	2	1	2	1	1	1	3	1	...	

5 rows × 26 columns

Figure 8.3: Label Encoder Mechanism

```
In [15]: data_encoded['Attrition'].value_counts()

Out[15]: 0    1233
         1     237
         Name: Attrition, dtype: int64
```

```
In [16]: #Plot to see distribution of age overall
         plt.rcParams["figure.figsize"] = [7,7]
         plt.hist(data_encoded['Age'], bins=np.arange(0,80,10), alpha=0.8, rwidth=0.9, color='blue')

Out[16]: (array([  0.,  17., 309., 622., 349., 168.,   5.]),
          array([ 0, 10, 20, 30, 40, 50, 60, 70]),
          <a list of 7 Patch objects>)
```

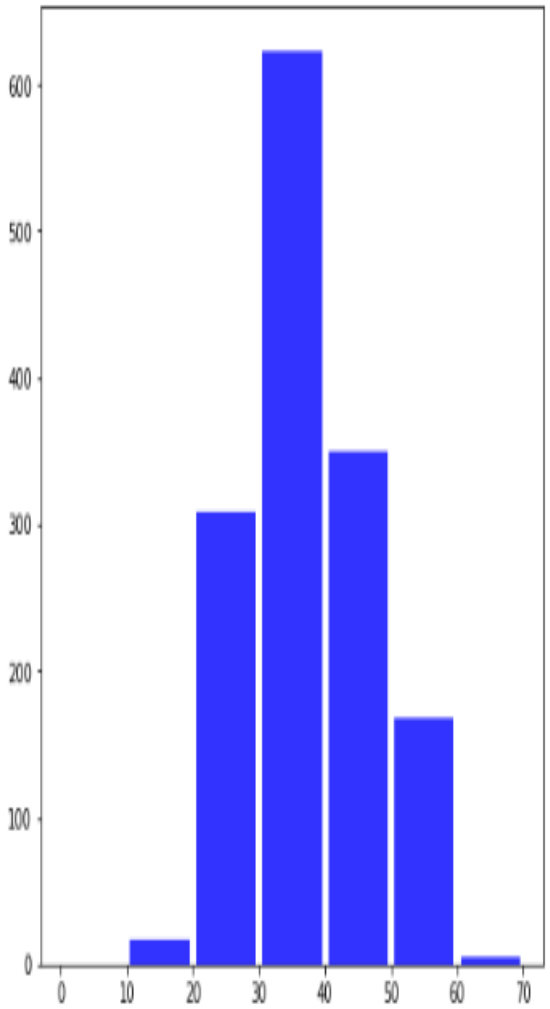
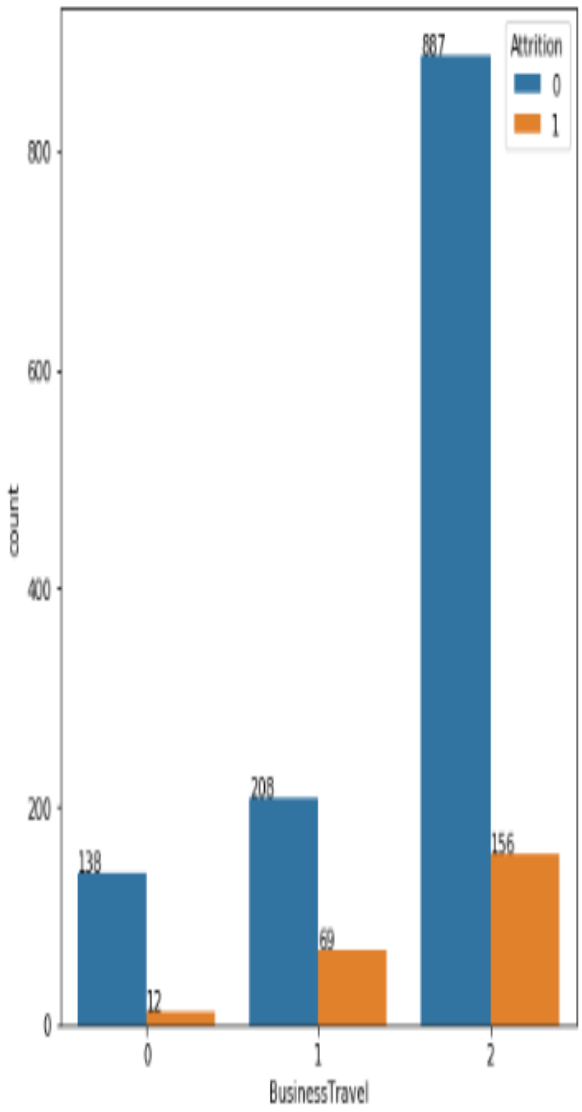


Figure 8.4: Histogram for Age Groups

```
In [17]: ax = sns.countplot(x="BusinessTravel", hue="Attrition", data=data_encoded)
         for p in ax.patches:
             ax.annotate('{}'.format(p.get_height()), (p.get_x(), p.get_height()+1))
```

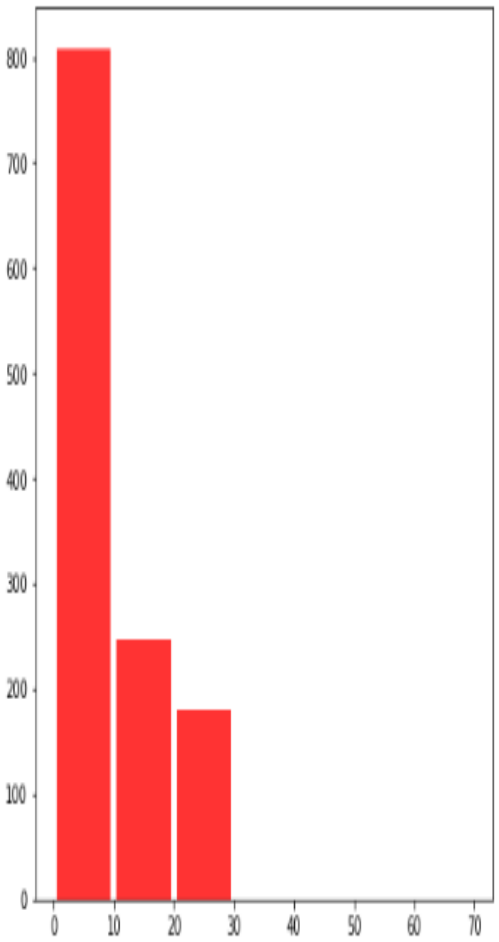



```
In [18]: positive_attrition_df = data_encoded.loc[data_encoded['Attrition'] == 1]
         negative_attrition_df = data_encoded.loc[data_encoded['Attrition'] == 0]
```

Figure 8.5: Count Plot for Business Travel

```
In [19]: plt.hist(negative_attrition_df['DistanceFromHome'], bins=np.arange(0,80,10), alpha=0.8, rwidth=0.9, color='red')
```

```
Out[19]: (array([807., 246., 180.,   0.,   0.,   0.,   0.]),
          array([ 0, 10, 20, 30, 40, 50, 60, 70]),
          <a list of 7 Patch objects>)
```




```
In [20]: df_age = data_encoded.copy(deep=True)
         df_age.loc[df_age['Age'] <= 20, 'Age'] = 0
         df_age.loc[(df_age['Age'] > 20) & (df_age['Age'] <= 30), 'Age'] = 1
         df_age.loc[(df_age['Age'] > 30) & (df_age['Age'] <= 40), 'Age'] = 2
         df_age.loc[(df_age['Age'] > 40) & (df_age['Age'] <= 50), 'Age'] = 3
         df_age.loc[(df_age['Age'] > 50), 'Age'] = 4
```

Figure 8.6: Histogram for Distance from Home

```
In [21]: df_age = pd.DataFrame({'count': df_age.groupby(["Gender", "Attrition"]).size()}).reset_index()
         df_age['Gender-attrition'] = df_age['Gender'].astype(str) + "-" + df_age['Attrition'].astype(str).map(str)

In [22]: df_age
```

Out[22]:

	Gender	Attrition	count	Gender-attrition
0	0	0	501	0-0
1	0	1	87	0-1
2	1	0	732	1-0
3	1	1	150	1-1

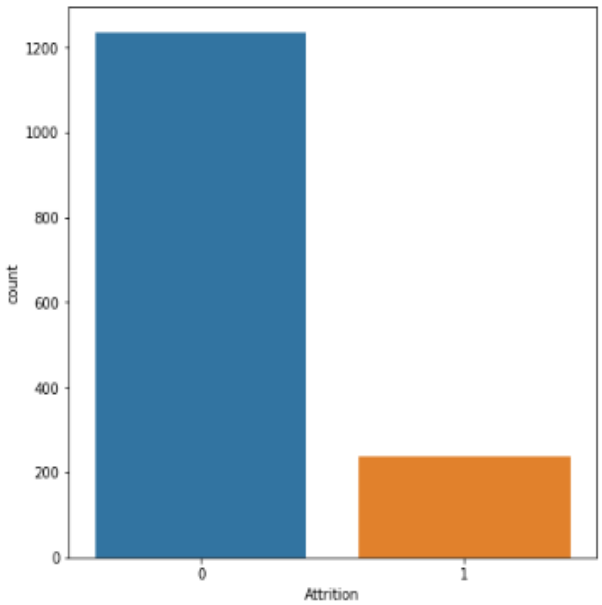
Here,

- Gender - 0 and Attrition - 0 ===> Female employees who will stay
- Gender - 0 and Attrition - 1 ===> Female employees who will leave
- Gender - 1 and Attrition - 0 ===> Male employees who will stay
- Gender - 1 and Attrition - 1 ===> Male employees who will leave

Figure 8.7: Group-By Function

```
In [23]: sns.countplot(x='Attrition',data=data_encoded)
```

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x19e758bfe80>




```
In [24]: shuffled_df = data_encoded.sample(frac=1,random_state=4)

         pos = shuffled_df.loc[shuffled_df['Attrition'] == 1]

         neg = shuffled_df.loc[shuffled_df['Attrition'] == 0].sample(n=500,random_state=42)
```

Figure 8.8: Count Plot for Attrition

```
In [25]: df=pd.concat([pos,neg])
         df.head()
```

Out[25]:

	Age	Attrition	BusinessTravel	Department	DistanceFromHome	Education	EnvironmentSatisfaction	Gender	JobInvolvement	JobLevel	...	PercentSalar
1153	18	1	1	2	3	2	2	0	3	1	...	
608	55	1	2	2	2	1	3	1	3	2	...	
293	26	1	2	2	4	4	4	1	2	2	...	
1452	50	1	1	2	1	4	2	1	3	2	...	
656	32	1	2	1	25	4	1	1	3	1	...	

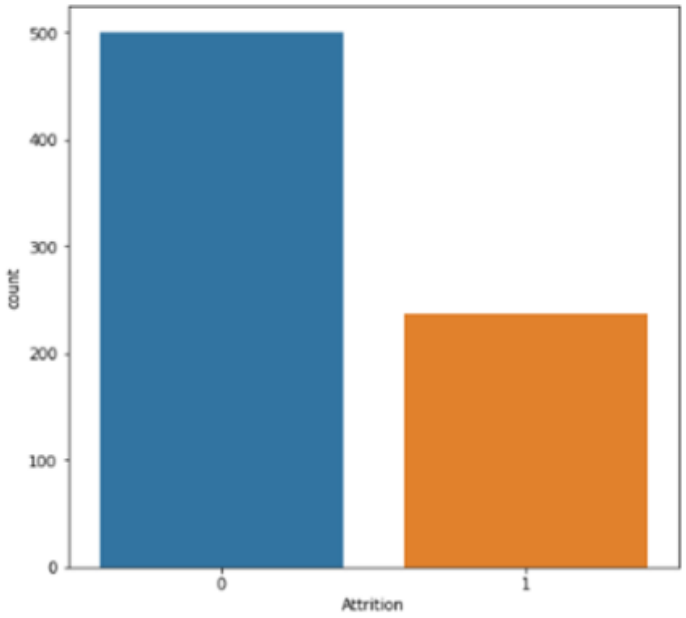
5 rows × 26 columns

```
In [26]: df.shape
```

Out[26]: (737, 26)

```
In [27]: sns.countplot(x='Attrition',data=df)
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x19e75967f60>




```
In [28]: df.to_csv('final.csv')
```

```
In [29]: X=df.iloc[:,df.columns !='Attrition']
         Y=df.iloc[:,df.columns =="Attrition"]
```

Figure 8.9: After Sampling is performed

In [30]:
```
X.head()
```

Out[30]:

	Age	BusinessTravel	Department	DistanceFromHome	Education	EnvironmentSatisfaction	Gender	JobInvolvement	JobLevel	JobRole	...	PercentSalar
1153	18	1	2	3	2	2	0	3	1	8	...	
608	55	2	2	2	1	3	1	3	2	7	...	
293	28	2	2	4	4	4	1	2	2	7	...	
1452	50	1	2	1	4	2	1	3	2	7	...	
656	32	2	1	25	4	1	1	3	1	2	...	

5 rows × 25 columns

In [31]:
```
Y.head()
```

Out[31]:

	Attrition
1153	1
608	1
293	1
1452	1
656	1

In [32]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,Y, test_size = 0.2, random_state=0)
```

In [33]:
```
X_train.shape
```

Out[33]: (589, 25)

In [34]:
```
X_test.shape
```

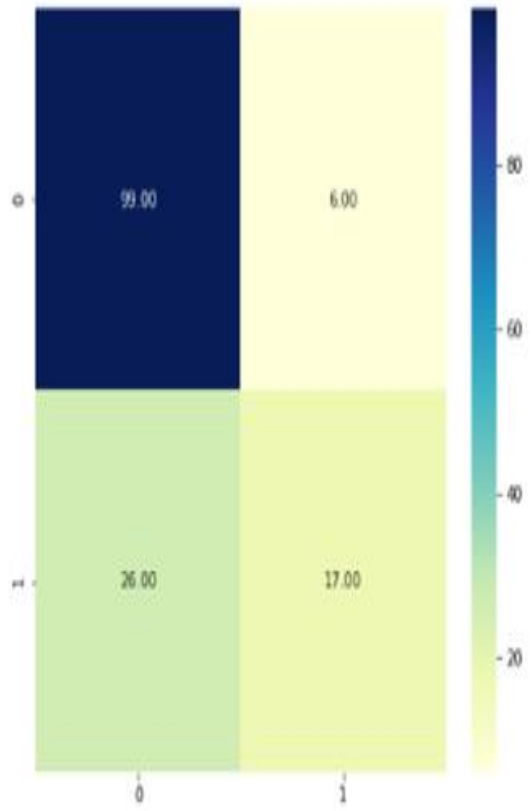
Out[34]: (148, 25)

Figure 8.10: Training and Test Data Split

```
In [35]: from sklearn.ensemble import RandomForestClassifier

In [36]: random_forest = RandomForestClassifier(n_estimators=100)

In [37]: random_forest.fit(X_train,y_train.values.ravel())

Out[37]: RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                                criterion='gini', max_depth=None, max_features='auto',
                                max_leaf_nodes=None, max_samples=None,
                                min_impurity_decrease=0.0, min_impurity_split=None,
                                min_samples_leaf=1, min_samples_split=2,
                                min_weight_fraction_leaf=0.0, n_estimators=100,
                                n_jobs=None, oob_score=False, random_state=None,
                                verbose=0, warm_start=False)

In [38]: y_pred = random_forest.predict(X_test)

In [39]: from sklearn.metrics import accuracy_score,confusion_matrix,classification_report

In [40]: rfacc=metrics.accuracy_score(y_test,y_pred)
         rfacc

Out[40]: 0.7837837837837838

In [41]: cnf_matrix = confusion_matrix(y_test,y_pred)
         cnf_matrix

Out[41]: array([[99,  6],
                [26, 17]], dtype=int64)
```

Figure 8.11: Random Forest Classifier

```
In [42]: labels = [0,1]
sns.heatmap(cnf_matrix, annot=True, cmap="YlGnBu", fmt=".2f", xticklabels=labels, yticklabels=labels)
plt.show()
```




```
In [43]: classifi1=classification_report(y_test,y_pred)
print(classifi1)
```

```
              precision    recall  f1-score   support

           0       0.79      0.94      0.86       105
           1       0.74      0.40      0.52        43

    accuracy                           0.78       148
   macro avg       0.77      0.67      0.69       148
weighted avg       0.78      0.78      0.76       148
```

Figure 8.12: Heatmap and Classification Report

```
In [44]: from sklearn import svm
         from sklearn import metrics

In [45]: model1=svm.SVC()

In [46]: model1.fit(X_train,y_train.values.ravel())

Out[46]: SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
             decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
             max_iter=-1, probability=False, random_state=None, shrinking=True,
             tol=0.001, verbose=False)

In [47]: xpredict=model1.predict(X_test)

In [48]: svm=metrics.accuracy_score(y_test,xpredict)
         print(svm)

         0.7094594594594594

In [49]: from sklearn.linear_model import LogisticRegression
         model2=LogisticRegression()

In [50]: model2.fit(X_train,y_train.values.ravel())

Out[50]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                            intercept_scaling=1, l1_ratio=None, max_iter=100,
                            multi_class='auto', n_jobs=None, penalty='l2',
                            random_state=None, solver='lbfgs', tol=0.0001, verbose=0,
                            warm_start=False)

In [51]: xpredict=model2.predict(X_test)

In [52]: lr=metrics.accuracy_score(y_test,xpredict)
         print(lr)

         0.75
```

Figure 8.13: SVM and Logistic Regression

In [53]:
```
from sklearn.neural_network import MLPClassifier
from sklearn.decomposition import KernelPCA
from imblearn.pipeline import make_pipeline
clf = MLPClassifier(solver='lbfgs', learning_rate='constant', activation='tanh')
kernel = KernelPCA()
pipeline = make_pipeline(kernel, clf)
pipeline.fit(X_train, y_train)
```

Out[53]:
```
Pipeline(memory=None,
         steps=[('kernelpca',
                 KernelPCA(alpha=1.0, coef0=1, copy_X=True, degree=3,
                           eigen_solver='auto', fit_inverse_transform=False,
                           gamma=None, kernel='linear', kernel_params=None,
                           max_iter=None, n_components=None, n_jobs=None,
                           random_state=None, remove_zero_eig=False, tol=0)),
                ('mlpclassifier',
                 MLPClassifier(activation='tanh', alpha=0.0001,
                               batch_size='..., beta_2=0.999,
                               early_stopping=False, epsilon=1e-08,
                               hidden_layer_sizes=(100,),
                               learning_rate='constant',
                               learning_rate_init=0.001, max_fun=15000,
                               max_iter=200, momentum=0.9, n_iter_no_change=10,
                               nesterovs_momentum=True, power_t=0.5,
                               random_state=None, shuffle=True, solver='lbfgs',
                               tol=0.0001, validation_fraction=0.1,
                               verbose=False, warm_start=False))],
         verbose=False)
```

In [54]:
```
xpredict4=pipeline.predict(X_test)
```

In [55]:
```
mlp=metrics.accuracy_score(y_test,xpredict4)
print(mlp)
```

```
0.6891891891891891
```

Figure 8.14: Multi-Layer Perceptron Classifier

```
In [56]: import matplotlib.pyplot as plt;
plt.rcdefaults()

objects = ('RandomForest','SupportVector',' LogisticRegression','MLPClassifier')
y_pos = np.arange(len(objects))
performance = [rfacc,svm,lr,mlp]

plt.bar(y_pos, performance, align='center', alpha=0.7)
plt.xticks(y_pos, objects)
plt.ylabel('Accuracy')
plt.title('RandomForest vs SVM vs LogisticRegression vs MLPClassifier')

plt.show()
```

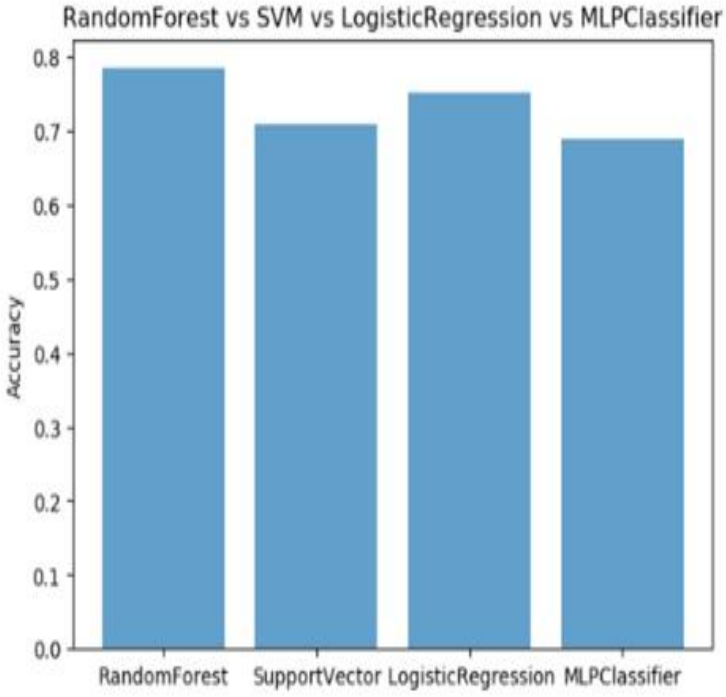


Figure 8.15: Plotting Accuracy in Graph

The accuracy scores returned are:

Random Forest Classifier – 0.7837

Support Vector Machine – 0.7094

Logistic Regression – 0.75

Multi-Layer Perceptron Classifier – 0.6891

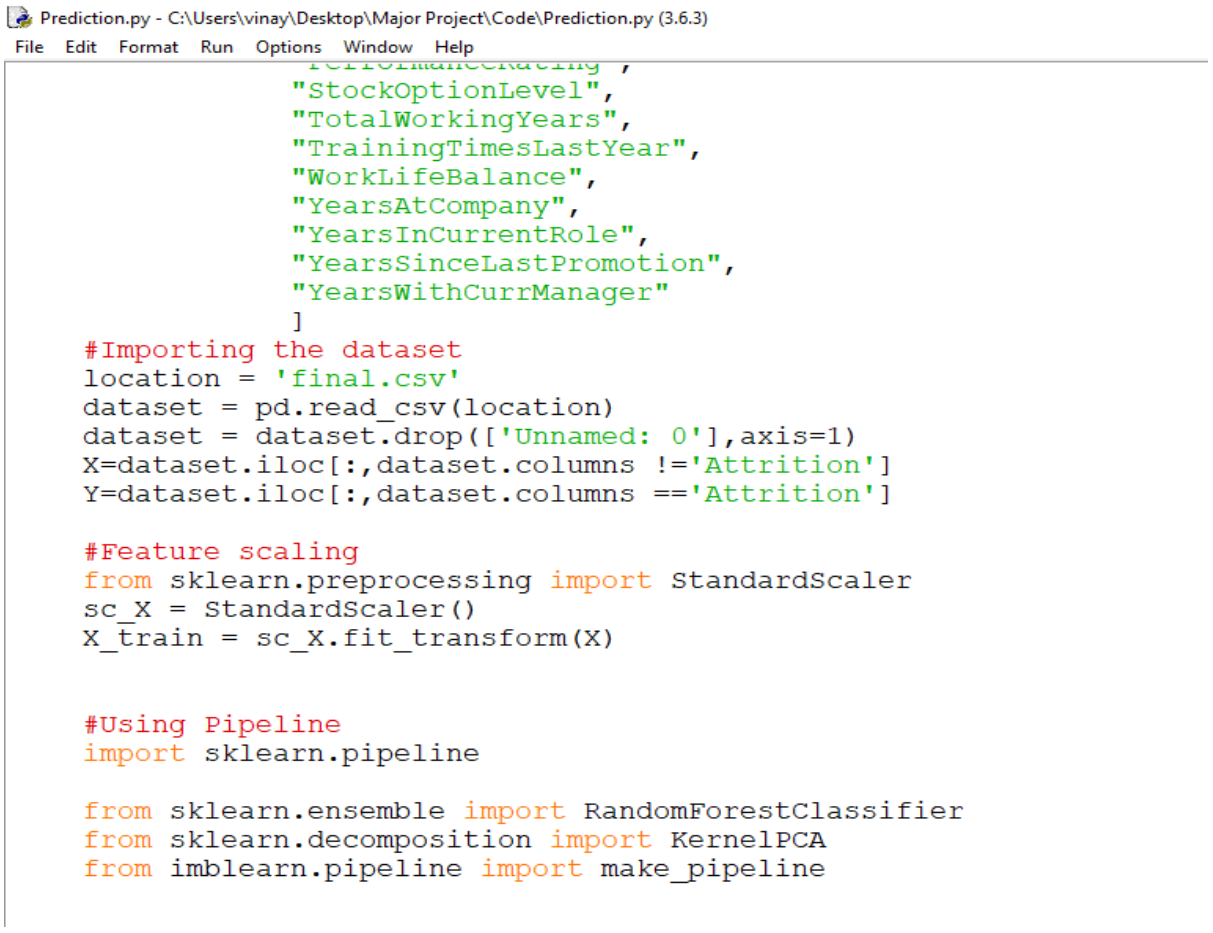
We got best accuracy using Random Forest Classifier Algorithm. So, we have to choose this algorithm in the remaining process of the code.

Prediction.py - C:\Users\vinay\Desktop\Major Project\Code\Prediction.py (3.6.3)

File Edit Format Run Options Window Help

```
if __name__ == "__main__":

    #Importing some libraries
    import numpy as np
    import pandas as pd
    import os
    #Getting rid of pesky warnings
    def warn(*args, **kwargs):
        pass
    import warnings
    warnings.warn = warn
    np.warnings.filterwarnings('ignore')

    column_names = [
                "Age",
                "BusinessTravel",
                "Department",
                "DistanceFromHome",
                "Education",
                "EnvironmentSatisfaction",
                "Gender",
                "JobInvolvement",
                "JobLevel",
                "JobRole",
                "JobSatisfaction",
                "MaritalStatus",
                "MonthlyIncome",
                "NumCompaniesWorked",
                "OverTime",
                "PercentSalaryHike",
                "PerformanceRating",
```

Figure 8.16: Python IDLE Code (1)




```
clf = RandomForestClassifier()
kernel = KernelPCA()

pipeline = make_pipeline(kernel, clf)
pipeline.fit(X,Y)

#User-input
v = []
for i in column_names[:]:
    v.append(input(i+": "))
answer = np.array(v)
answer = answer.reshape(1,-1)
answer = sc_X.transform(answer)
print ("Predicts:"+ str(pipeline.predict(answer)))
```

Figure 8.17: Python IDLE Code (2)

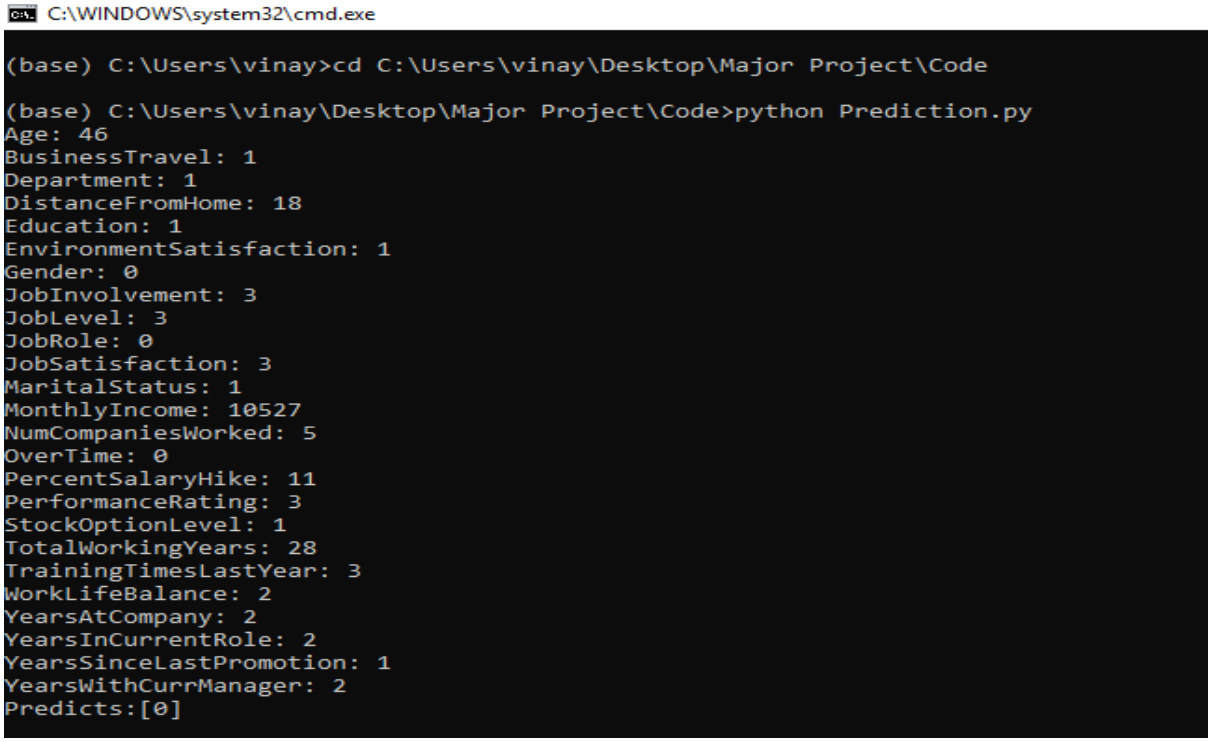


Figure 8.18: Employee Stays

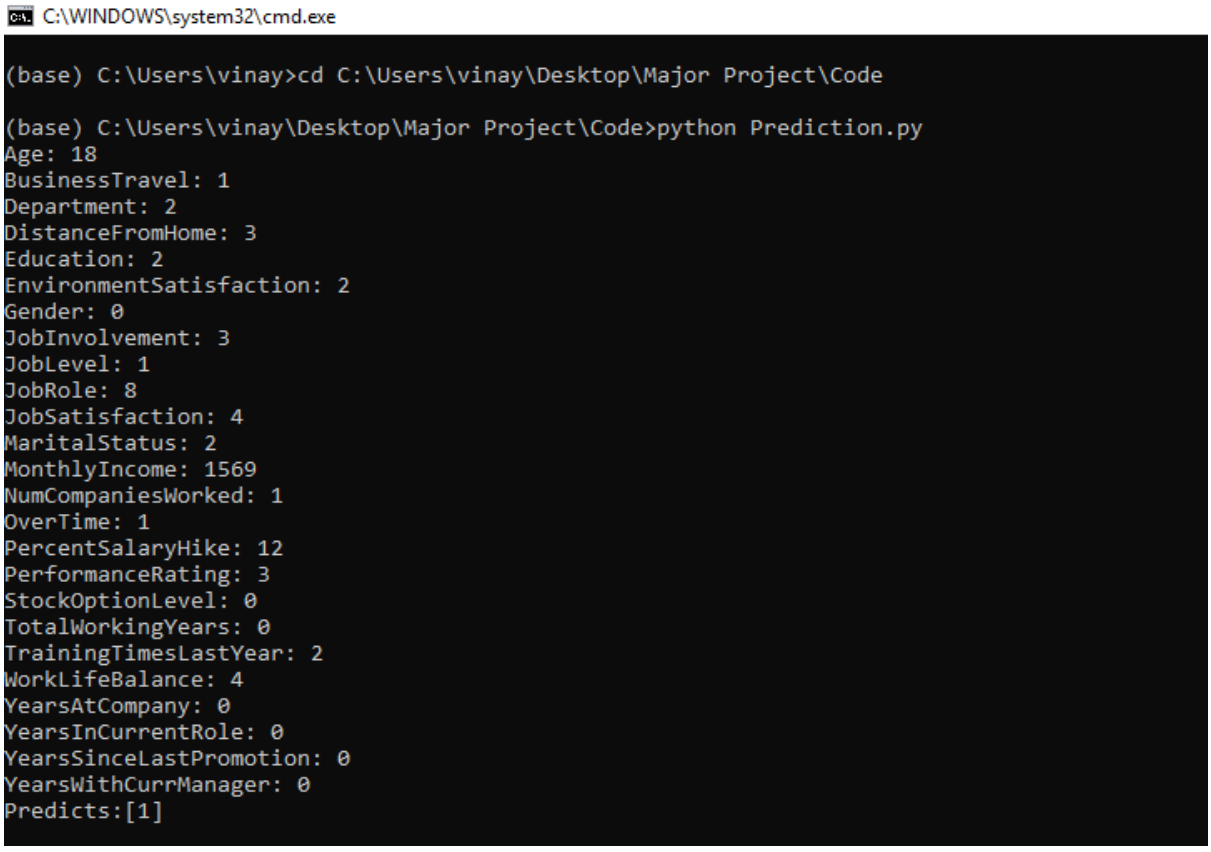


Figure 8.19: Employee Leaves

(9) FUTURE SCOPE OF THE PROJECT

- AI Fairness 360(AIF), a comprehensive open source toolkit of metrics to check for unwanted bias in datasets and machine learning models. This model could be used to eliminate the bias in the dataset.

- The model developed will be able to predict whether an employee will stay or not this will help company to know the status of an employee in advance and take necessary actions to prevent loss that will incur.

- The findings of the study are subjected to bias and prejudice of the respondent's time factor can be considered as main limitations the findings of the study are solely depend on the information provided by respondents.

- The accuracy of findings is limited by the accuracy of statistical tools used for analysis. Findings of the research may change due to area demography, age condition of economy etc.

(10) CONCLUSION

The main objective of the organization is to earn profit and to earn profit the employer should concentrate in retaining talents and concentrate in making them stick to the organization for the long run. Employees are the assets of the organization. Hence it is important for the employers to minimize the attrition rate and help in both individual as well as organizational growth. Thus, Organizations should create an environment that fosters ample growth opportunities, appreciation for the work accomplished and a friendly cooperative atmosphere that makes an employee feel connected in every respect to the organization. Retention plans are an inexpensive way of enhancing workplace productivity and engaging employees emotionally. Proficient employees keep the quality up and business operations run smoothly along with the cost saving in the longer run. It can be concluded that the following are the major reasons for employee turnover

- Opportunity for growth and promotion outside
- Compensation
- Working conditions
- Work timings/shifts
- Relationship with managers
- Location of the organisation
- Work load

(11) REFERENCES

- Understanding Machine Learning: From Theory to Algorithms by Shai Shalev-Schwartz and Shai Ben-David
- Scikit-Learn Tutorial: Statistical-Learning for Scientific Data Processing by Andreas Mueller
- Machine Learning (An Algorithmic Perspective) by Stephen Marsland
- A Brief Introduction to Neural Networks by David Kriesel
- https://www.tutorialspoint.com/machine_learning/index.htm
- https://en.wikipedia.org/wiki/Machine_learning
- https://developer.ibm.com/patterns/data-science-life-cycle-in-action-to-solve-employee-attrition-problem/
- https://www.kaggle.com/pavansubhasht/ibm-hr-analytics-attrition-dataset
- https://towardsdatascience.com/a-beginners-guide-to-the-data-science-pipeline-a4904b2d8ad3
- https://dzone.com/articles/overview-of-the-data-science-pipeline