A PROJECT REPORT

on

"AI to classify LinkedIn experiences"

Submitted to

CY Tech

In Partial Fulfillment of the Requirement for the Award of

BACHELOR'S DEGREE IN

INFORMATION TECHNOLOGY

BY

Little Beauty Bisoyi 1906335

UNDER THE GUIDANCE OF

Mr. Thomas DANGUILHEN

CY Cergy Paris UNIVERSITE`

33 Bd du Port, 95000 Cergy
Île-de-France, France

CY Cergy Paris UNIVERSITE`

33 Bd du Port, 95000 Cergy
Île-de-France, France

CERTIFICATE

This is certify that the project entitled

"AI to classify LinkedIn experiences"

submitted by

Little Beauty Bisoyi

1906335

is a record of bonafide work carried out by them, in the partial fulfilment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Engineering OR Information Technology) at CY Tech, CY Cergy Paris UNIVERSITE. This work is done during year 2023-2024, under our guidance.

Date: 12/06/2023

Mr. Thomas DANGUILHEN

Project Guide

Acknowledgments

I would like to express my sincere gratitude to my mentor, Thomas Danguilhen, for his invaluable guidance, support, and expertise throughout the duration of this project. His deep knowledge and passion for the subject matter have been instrumental in shaping the direction of this research. His constant encouragement, insightful feedback, and unwavering commitment to excellence have been truly inspiring.

I would also like to extend my appreciation to CY Tech, the college where this research was conducted. The college's conducive environment and resources have provided a platform for exploration and learning. The collaborative atmosphere and access to cutting-edge technologies have greatly enhanced the quality of this project.

I am also thankful to the faculty members and staff of CY Tech for their assistance, encouragement, and for creating a stimulating academic atmosphere that nurtures innovation and growth.

Furthermore, I would like to acknowledge the contributions of my fellow mates who have provided valuable insights and suggestions throughout the course of this work. Their discussions and inputs have been instrumental in shaping and refining the ideas presented in this project.

Without the support and guidance of all these individuals and institutions, this project would not have been possible. Their contributions have played a pivotal role in the development and accomplishment of this work, and for that, I am truly grateful.

Little Beauty Bisoyi - 1906335

ABSTRACT

The increasing popularity of professional networking platforms such as LinkedIn has led to a significant growth in the amount of user-generated content related to work experiences. The classification and analysis of these experiences can provide valuable insights for individuals seeking employment opportunities, recruiters looking for suitable candidates, and organizations aiming to understand industry trends. However, manually sifting through and categorizing large volumes of LinkedIn experiences is a time-consuming and labor-intensive task.

This paper presents an AI-based approach to automatically classify LinkedIn experiences using natural language processing and machine learning techniques.
The system utilizes a combination of text pre-processing, feature extraction, and supervised learning algorithms to train a model capable of accurately categorizing LinkedIn experiences into relevant domains such as job titles, industries, and job responsibilities. We employ techniques such as tokenization, word embedding, and feature selection to enhance the representation of textual data and capture the semantic meaning of experience descriptions.

To evaluate the performance of our AI model, we collected a diverse dataset of LinkedIn experiences, including a wide range of industries, job titles, and descriptions. We conducted experiments using various machine learning algorithms and evaluated the classification accuracy, precision, recall, and F1-score. The results demonstrate the effectiveness of our AI system in accurately categorizing LinkedIn experiences, outperforming traditional rule-based methods and achieving high levels of accuracy and efficiency.

The implications of our research extend beyond the realm of individual job seekers and recruiters. By automating the classification of LinkedIn experiences, organizations can gain valuable insights into emerging industry trends, identify skill gaps, and make informed decisions regarding talent acquisition and workforce planning.

Overall, this paper contributes to the advancement of AI applications in the context of professional networking platforms by introducing an effective and scalable solution for classifying LinkedIn experiences. The proposed system has the potential to revolutionize the way we analyze and leverage the vast amount of user-generated content on LinkedIn, ultimately facilitating more efficient and informed decision-making processes in the job market.

Contents

Introduction

LinkedIn has emerged as a prominent platform for professionals to showcase their skills, experiences, and career aspirations. With millions of users worldwide, the platform has become a treasure trove of valuable information about individuals' work histories, job titles, industry affiliations, and job responsibilities. However, the sheer volume of user-generated content on LinkedIn poses a significant challenge for individuals, recruiters, and organizations seeking to make sense of this wealth of data.

Manual classification and analysis of LinkedIn experiences can be an arduous and time-consuming task. To address these challenges, artificial intelligence (AI) techniques offer a promising solution. Leveraging natural language processing (NLP) and machine learning (ML) algorithms, AI can automate the process of classifying LinkedIn experiences, enabling faster and more accurate analysis of user profiles. By automatically extracting relevant information from textual descriptions, AI can categorize experiences into domains such as job titles, industries, and job responsibilities, providing users with valuable insights and helping recruiters identify suitable candidates.

The classification of LinkedIn experiences has significant implications for various stakeholders. Job seekers can benefit from AI-powered systems that analyze their profiles and provide personalized recommendations for career development and networking opportunities. Recruiters and hiring managers can leverage AI to efficiently search and filter through a vast pool of candidates, narrowing down their selection to those most aligned with their requirements. Additionally, organizations can gain valuable insights into industry trends, skill gaps, and talent acquisition strategies by analyzing the aggregated data from LinkedIn experiences.

This paper aims to explore the use of AI techniques for the classification of LinkedIn experiences. We propose a novel approach that combines NLP and ML algorithms to automatically categorize experiences based on their textual descriptions. By training a model on a diverse dataset of LinkedIn profiles, we evaluate the performance and effectiveness of our proposed system.

Overall, the integration of AI into the classification of LinkedIn experiences has the potential to revolutionize the way professionals connect, recruiters identify talent, and organizations gain insights from user-generated content. By automating this process, AI can enhance efficiency, accuracy, and the overall user experience, enabling individuals and organizations to make more informed decisions in the ever-evolving job market.

Objectives

The objective of developing an AI to classify LinkedIn experiences can be broken :

1. Data Collection: Gather a substantial amount of LinkedIn profiles and their corresponding experiences. This includes obtaining data from various industries, job roles, and levels of experience to ensure diversity and representativeness.

2. Data Preprocessing: Clean and preprocess the collected data to remove irrelevant information, handle missing values, standardize formats, and ensure data quality. This step is crucial for preparing the data for training the AI model.

3. Feature Extraction: Extract meaningful features from the LinkedIn experiences to represent the key aspects of each job role or experience. This could involve techniques such as natural language processing (NLP) to analyze the text, identifying keywords, extracting job titles, and identifying relevant skills.

4. Labeling and Annotation: Create a labeled dataset where each LinkedIn experience is assigned relevant categories or tags. This requires human experts to manually label a subset of the data to serve as a training set for the AI model.

5. Model Training: Develop and train a classification model using machine learning or deep learning techniques. The model should learn from the labeled dataset and be able to accurately classify LinkedIn experiences into appropriate categories or tags.

6. Evaluation and Validation: Assess the performance of the trained model using evaluation metrics such as accuracy, precision, recall, and F1-score. Validate the model's effectiveness by comparing its predictions with human-labeled data or expert judgments.

7. Iterative Refinement: Continuously refine the model based on feedback and improve its performance. This could involve incorporating new data, adjusting the model architecture, fine-tuning hyperparameters, or exploring different algorithms.

8. Deployment and Integration: Integrate the trained model into a user-friendly application or system that allows users to input their LinkedIn experiences and receive accurate classifications or tags. Ensure the model is scalable, efficient, and can handle real-time classification requests.

9. User Feedback and Improvement: Gather user feedback and monitor the performance of the deployed AI system. Incorporate user suggestions and continuously update the model to improve its accuracy and adapt to evolving patterns in LinkedIn experiences.

Technology used

1. Basic of Python

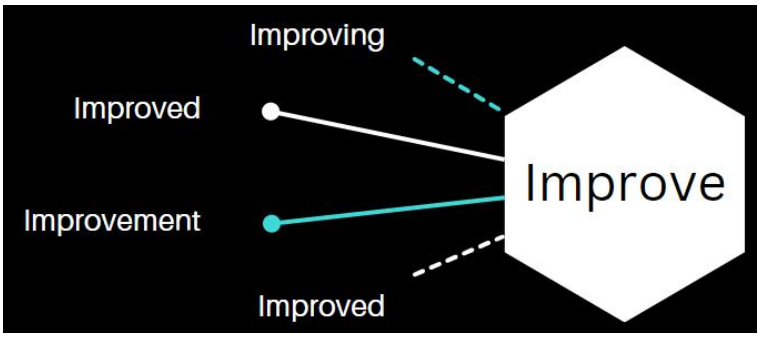
Python is a Multi-Purpose programming language. It is used for developing GUI (Graphical User Interfaces), various scripting purposes, creating backend applications, web scraping and various other things. It is an Interpreted Language, that is, it is executed in a sequential manner and does not need to be compiled before it is executed. It is a strongly and dynamically typed programming language which is extendable and portable. It can be used to combine various programming languages together to work cohesively as one distinct entity. In addition to that, Python is also a free and open source programming language which means that it is free to use and everyone can contribute to its development.

2. Machine Learning

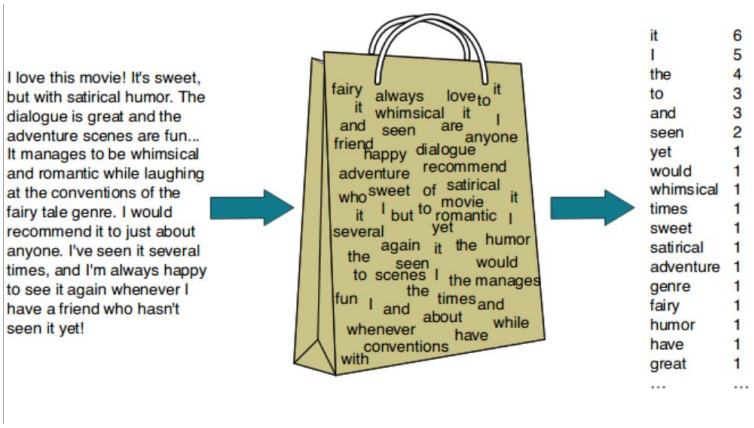
Machine Learning is a field of artificial intelligence built on the principle that computers can learn from data, recognize patterns, and make judgments with little or no human input.
Machine learning now is not the same as machine learning in the past, thanks to advances in computer technology. It was inspired by pattern recognition and the idea that computers may learn without being taught to execute certain tasks; artificial intelligence researchers sought to investigate if computers could learn from data. The iterative feature of machine learning is crucial because models may evolve autonomously as they are exposed to fresh data. They use past computations to provide consistent, repeatable judgments and outcomes. It's a science that's not new, but it's gaining new traction. While many machine learning techniques have been known for a while, the capacity to apply difficult mathematical computations to large amounts of data automatically – again and again, quicker and quicker – is a relatively new phenomenon.

Important concepts used

Lemmatization : Lemmatization is the process of reducing words to their base or dictionary form, known as the lemma.



Bag of Words : The "bag of words" is a text representation model that disregards the order and structure of words, treating a document as a collection of individual words.



3. Deep Learning

Deep learning is a powerful and popular approach to artificial intelligence that has revolutionized many fields, including computer vision, natural language processing, and speech recognition. It is a sub field of machine learning that focuses on training neural networks with multiple layers to learn complex patterns and make predictions or classifications.

In simple terms, deep learning is like teaching a computer to think and learn in a similar way to humans. Just like our brain consists of interconnected neurons that process information, deep learning models are designed with layers of artificial neurons called "neural networks." These networks are trained using large amounts of data, allowing them to recognize and understand patterns, features, and relationships within the data.

Important concepts used

Transformers : Transformers are a type of deep learning model architecture that uses self-attention mechanisms to process sequential data and achieve state-of-the-art performance in various natural language processing tasks.

BERT Model : BERT (Bidirectional Encoder Representations from Transformers) is a pre-trained deep learning model that uses a transformer architecture for various natural language processing tasks.

Problem Statement

We have a large number of LinkedIn profiles, and each profile has on average 5 previous experiences. We want to find a method to automatically classify the previous experiences as AI or non-AI.

Softwares Used : Google Colab, Jupyter notebook

Languages Used: Python, shell scripting

Methodology

1. Importing libarraies

2. Loading and reading the dataset

3. Data cleaning

   3.1.Display the Null values percentage against every columns (compare to the total number of records)
   3.2.Removing Rows with label NAN
   3.3.Replace None in Description column with empty string
   3.4.Concatenating columns jobTitle and description as jobDescription
   3.5.Converting jobDescription column into lowercase
   3.6.Removing stopwords
   3.7.Removing unecessary columns

4. Label Encoding

5. EDA

6. Feature selection and feature engineering
   6.1. Lemmatization
   6.2. Creating bag of words

7.Splitting

8. Modelling
   7.1. Logistic regression
   7.2. K-Nearest Neighbour
   7.3. Decision Tree Classifier
   7.4. Random Forest Classifier

9.Ploting the Classification Accuracy for every Algo

10. Checking for Overfitting
    10.1. Checking the Mean Absolute Error
    10.2. k-fold cross-validation

Methodology

1. Importing different Libraries

We have imported the libraries such pandas, pickle, numpy, sklearn, nltk, seaborn, matplotlib. The resaon behind importing each of the libraries is being explained below in the steps where they are imported.

2. Loading and reading the dataset

Loading the pickle dataset into Google Colab. Converting into dataframe df and reading it.

After reading the dataframe we found that there are 9000 rows and 8 columns in the dataframe. So the shape of the dataframe is (9000,8).

The columns are:
1. companyName
2. companyUrl
3. jobTitle
4. dateRange
5. Location
6. Description
7. logoUrl
8. Label

Then we did df.head() to get better understanding of the dataframe. It displays the first 5 rows and all the columns of the dataframe.

Then I perofrmed df.describe() for obtaining a quick statistical overview of numerical columns in a DataFrame. It provides important summary statistics that aid in understanding the central tendency, dispersion, outliers, and data distribution. This information forms the basis for further data exploration, visualization, and analysis.

3. Data Cleaning

A. Display the Null values percentage against every columns (compare to the total number of records)

Then we find the null value percentage against each column. The main purpose behind this step is find the presence of missing values in the dataset that can indicate potential data quality issue. Here we iterate over each column in the DataFrame df, calculate the percentage of null values in each column, and print the column name and the corresponding percentage if the percentage is non-zero. This code can be useful for identifying columns with missing data in your DataFrame.

B. Removing Rows with label NAN

This step performs the removal of rows with missing values specifically in the 'label' column of the DataFrame `df`. We generally perform this step as missing values can create gaps in the dataset.
The DataFrame `df` will have any rows with missing values in the 'label' column eliminated.

After dropping the rows with missing label columns the shape of the dataframe is (5651,8), the shape of our new dataframe.

C. Replacing 'None' in Description column with an empty string

Here in this step we replace the 'None' values in the description column with empty strings. We use the fillna method to perform this operation. It is generally done in data processing step when we are working with text or string data.

D. Concatenating columns jobTitle and description as jobDescription

The line of code `df["jobDescription"] = df[['jobTitle', 'description']].agg(' '.join, axis=1)` performs the concatenation of the 'jobTitle' and 'description' columns in the DataFrame `df` to create a new column named 'jobDescription'.

After executing this code, a new column named 'jobDescription' will be added to the DataFrame `df`, containing the concatenation of the 'jobTitle' and 'description' columns with a space as the separator. The resulting DataFrame will have the original columns intact, along with the new 'jobDescription' column.

We perform this function as it will be easier for the machine to work on the single column then working on two similar column.

E. Converting jobDescription column into lowercase

In the context of the project, this code is used to convert the text in the 'jobDescription' column to lowercase. It helps to ensure consistency and standardization in the text data, as lowercase letters are commonly used in natural language processing and text analysis tasks.

By converting all text to lowercase, you eliminate the variations caused by uppercase and lowercase letters. This ensures that the same word or phrase is treated as identical regardless of its capitalization. For example, "Hello," "hello," and "HELLO" will all be transformed to "hello," making it easier to compare, match, or analyze the text.

By calling `df['jobDescription'].apply(low)`, each string value in the 'jobDescription' column will be transformed to lowercase, and the resulting modified column will be assigned back to the 'jobDescription' column of the DataFrame `df`. This updates the 'jobDescription' column with the lowercase text.

F. Removing stop words

We will be removing the stopwords as they don’t carry any significant meaning or contribute much to the overall understanding of the text.

nltk.download('stopwords') is used to download the stopwords corpus from NLTK (Natural Language Toolkit).nltk.download('stopwords'), we can import the stopwords and use them to remove these common words from your text data, which can help improve the accuracy and relevance of your analysis or model.

Here we first of all retrieves the set of French and English stopwords from the NLTK stopwords corpus.This code retrieves the set of English stopwords from the NLTK stopwords corpus.

After removing the stopwords from ‘JobDescription’ column we create a new column’’jobDescription_without_stopwords’.

G. Removing unecessary columns

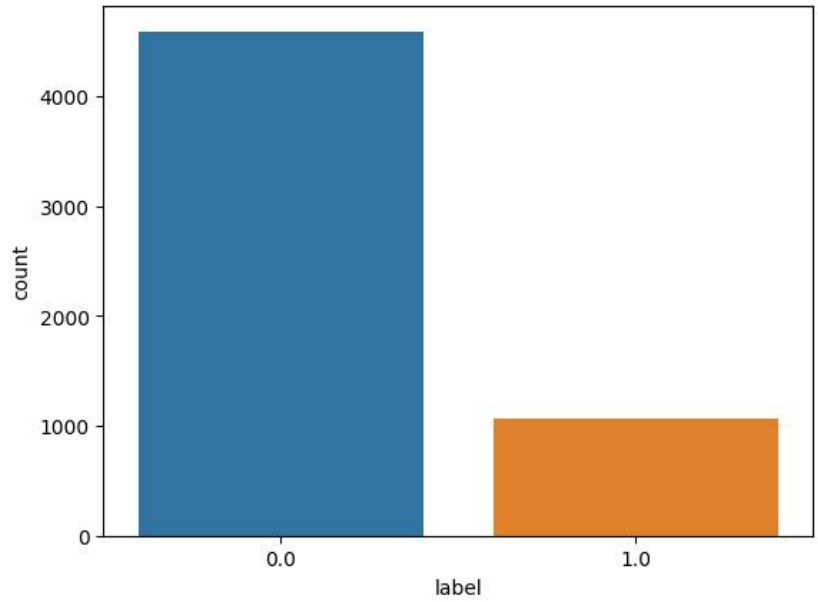
Columns 'companyUrl', 'dateRange', 'location', 'logoUrl', 'jobDescription' are removed from the DataFrame `df`. After executing this code, the DataFrame `df` will no longer have the columns 'companyUrl', 'dateRange', 'location', 'logoUrl', and 'jobDescription'. These columns are effectively removed from the DataFrame, and the modified DataFrame is assigned back to the variable `df`.

H. Relabeling the column ‘Label’

The given lines of code perform value replacement in the 'label' column of the DataFrame `df`. After executing these lines of code, the 'label' column in the DataFrame `df` will have undergone value replacement.

The values 0.0, 1.0, and 2.0 will be replaced with 0.0, while the values 3.0 and 4.0 will be replaced with 1.0. This type of value replacement is useful for mapping categorical or numerical values to a specific set of values that align with the desired data representation or analysis requirements.

4. Exploratory data analysis



The provided code generates a count plot to visualize the distribution of the 'label' column in the DataFrame `df`. After executing this code, we will see a count plot that provides a visual representation of the distribution of the 'label' column in the DataFrame `df`. It helps to understand the balance or imbalance of different categories or values in the 'label' column and can provide insights into the dataset's composition.

The plot shows that the dataset is a imbalanced dataset.

5. Feature Engineering

Feature engineering is the process of transforming raw data into meaningful and informative features that enhance the performance and interpret ability of machine learning models.

Before moving to 1st step of Feature engineering we will be selecting our X and Y column.

A. Lemmatization

The provided code snippet demonstrates the process of lemmatizing a text column using the NLTK library in Python. Lemmatization is a technique used in natural language processing to reduce words to their base or root form, which helps in normalizing text and capturing the underlying meaning of words.

First, the code imports the necessary modules from the NLTK library and downloads the required resources for lemmatization, namely the WordNet lexical database and the Punkt tokenizer. These resources provide the necessary information and rules for performing lemmatization.

Next, an instance of the WordNetLemmatizer class is initialized as the lemmatizer object, which will be used for lemmatizing words.

The code defines a function called "lemmatize_sentence" that takes a sentence as input. Inside the function, the sentence is tokenized into individual words using the nltk.word_tokenize() function. Then, a loop iterates through each token and applies lemmatization using the lemmatizer.lemmatize() method. The lemmatized tokens are appended to a list. After processing all the tokens, the lemmatized tokens are joined back into a sentence using the ' '.join() method.

Finally, the lemmatize_sentence() function is applied to a specific column in the DataFrame called "jobDescription_without_stopwords" using the DataFrame's apply() method. This lemmatizes the text in that column and updates the DataFrame with the lemmatized version.

In summary, the code snippet utilizes the NLTK library to perform lemmatization on a text column. It downloads the necessary resources, initializes a lemmatizer object, defines a lemmatization function, and applies the function to a specific column in a DataFrame, resulting in the lemmatization of the text data.

B. Creating Bag of Words

Creating a bag of words (BoW) is a common technique in natural language processing for representing text data as numerical features that machine learning models can process.
First of all we will perform tokenization: The text data is split into individual words or tokens. This process breaks down the text into its constituent units, such as words or characters, depending on the chosen tokenization strategy.

Then it create an unique vocabulary is created by collecting all the unique tokens from the text data. Each token becomes a distinct feature in the BoW representation.

For each document or text instance, the number of occurrences of each token in the vocabulary is counted. This creates a numerical vector representation for each document, where each element represents the count of a specific token in the document.

The counts are organized into a matrix, where each row corresponds to a document and each column corresponds to a token in the vocabulary. This matrix is often referred to as the BoW matrix or the term-document matrix.

The resulting BoW representation captures the frequency or presence/absence information of the tokens in each document. This representation is commonly used as input for various machine learning algorithms to perform tasks such as text classification, sentiment analysis, or topic modeling.

6. Splitting

Then we will be splitting the dataset into train a test set. We will be giving the test size as 0.25. The shape of X_train is (3673,120), X_test is (1978, 120), Y_train is (3673,1), and Y_test is (1978,1).

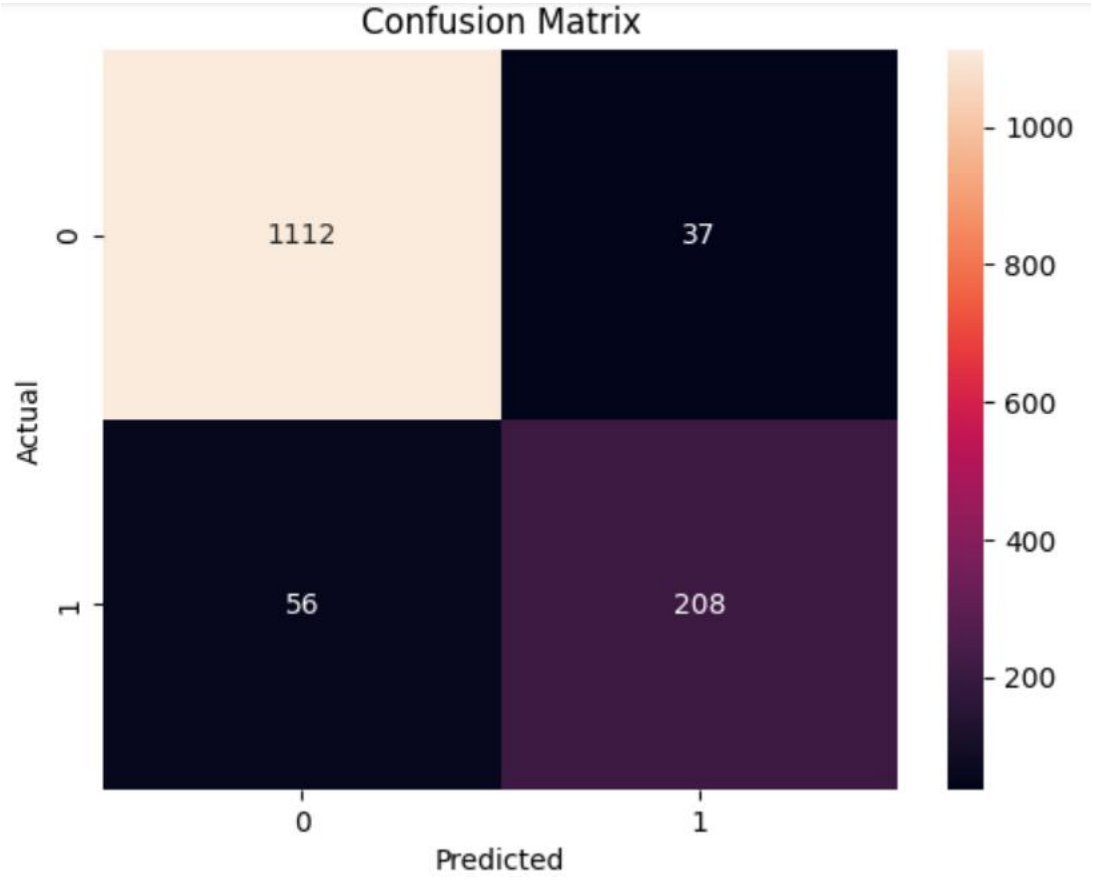
7. Modelling

In machine learning, modeling refers to the process of creating and training a mathematical or computational algorithm that can learn patterns and make predictions from data. It involves selecting an appropriate algorithm or model architecture, defining input features, and optimizing model parameters using training data. The goal of modeling in machine learning is to develop a model that can generalize well to unseen data and accurately predict outcomes or make decisions based on input features. The model is evaluated using performance metrics, and iterative refinement may be performed to improve its accuracy and generalization capabilities. Once the model is trained, it can be used for predictions or inferences on new, unseen data.

A. Logistic Regression

Definition: Logistic regression is a statistical model used to predict the probability of a binary outcome based on input variables.

The code defines a function called `logisticsRegression` that trains a logistic regression model, makes predictions, evaluates the model's performance using the `evalClassModel` function, and stores the accuracy score and MSE in the `methodDict` dictionary.

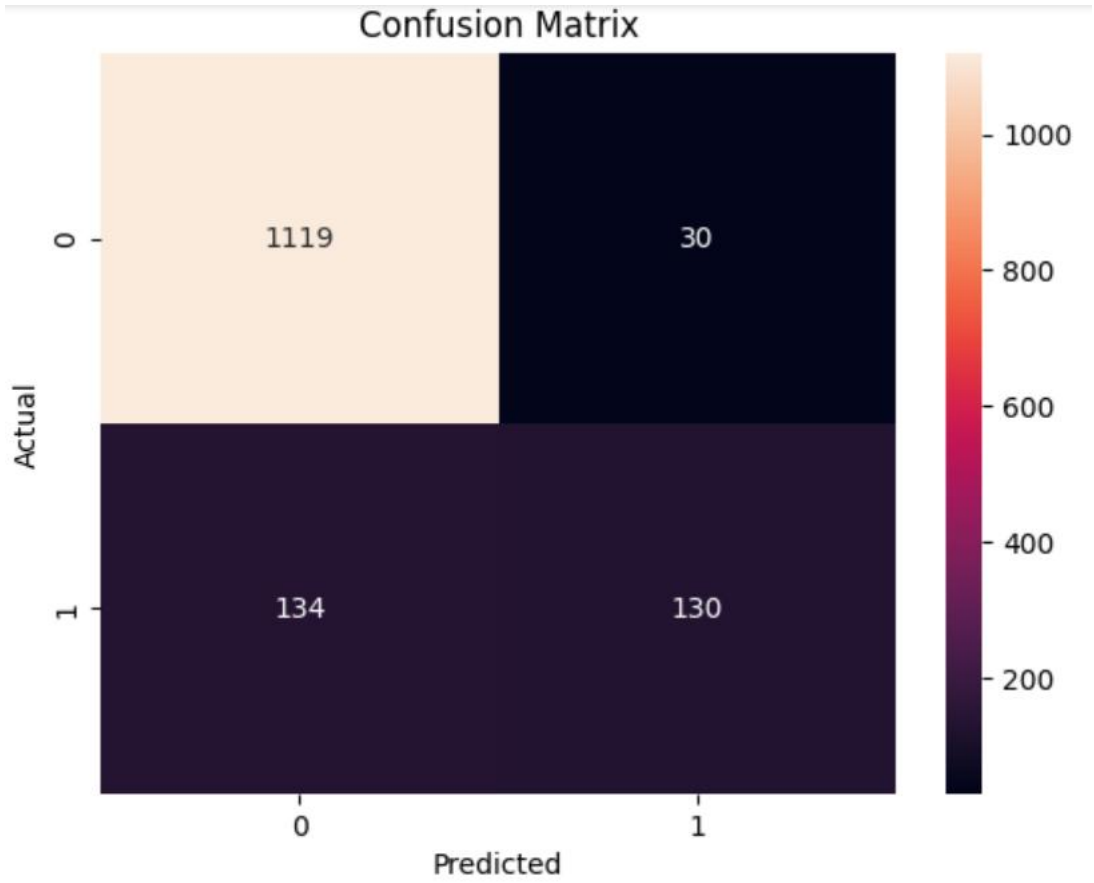


Classification Accuracy: 0.9342770475227502
Classification Error: 0.0657229524772498
False Positive Rate: 0.029850746268656716
Precision: 0.8571428571428571
F1 score :- 0.8158640226628895
Mean Squared Error: 0.06572295247724974

B. K-Nearest Neighbour

Definition : K-Nearest Neighbors (KNN) is a simple and intuitive machine learning algorithm that classifies a data point based on the majority vote of its k nearest neighbors in the feature space.

The code defines a function called `Knn` that trains a KNN classifier with different values of `n_neighbors` and `weights`, makes predictions, evaluates the model's performance using the `evalClassModel` function, and stores the accuracy score and MSE in the `methodDict` dictionary.

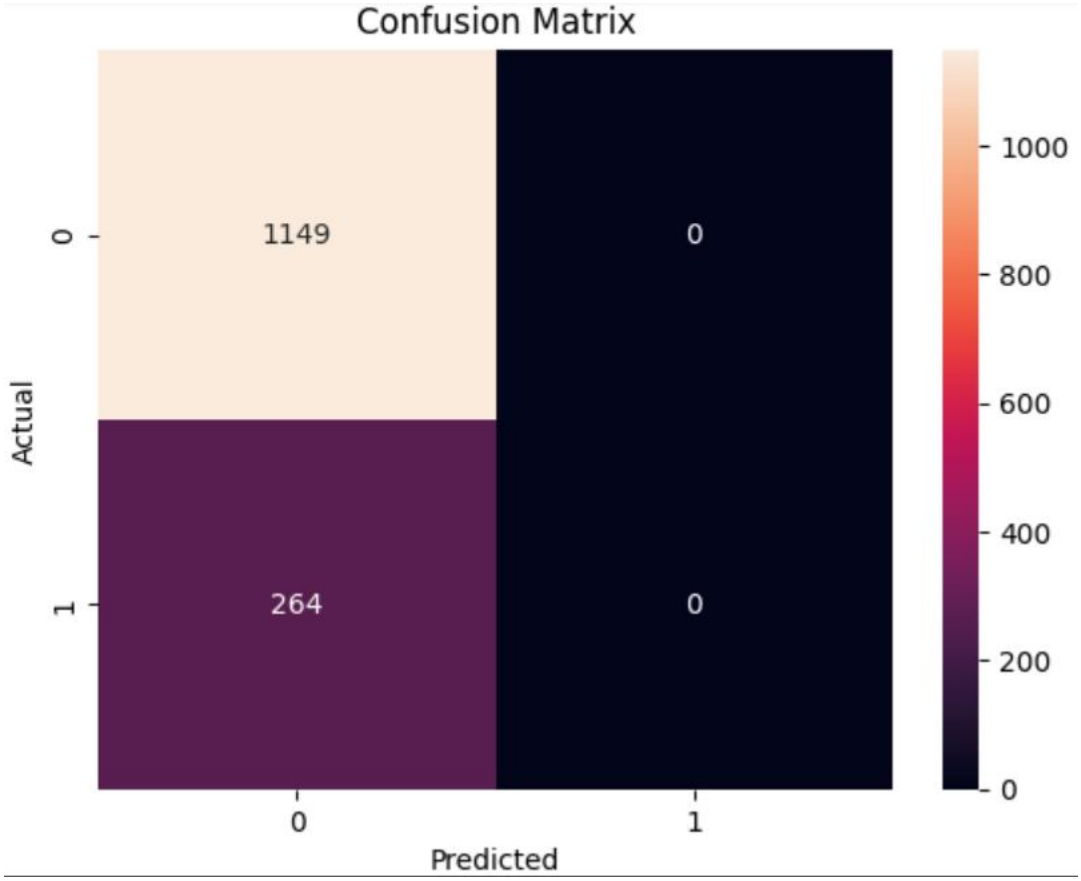


Classification Accuracy: 0.8892821031344793
Classification Error: 0.11071789686552069
False Positive Rate: 0.021766169154228857
Precision: 0.8416289592760181
F1 score :- 0.6294416243654822
Mean Squared Error: 0.11071789686552073

C. Decision Tree Classifier

The instance of the DecisionTreeClassifier class is created and assigned to the variable tree. This will be used to train and predict with a decision tree model.

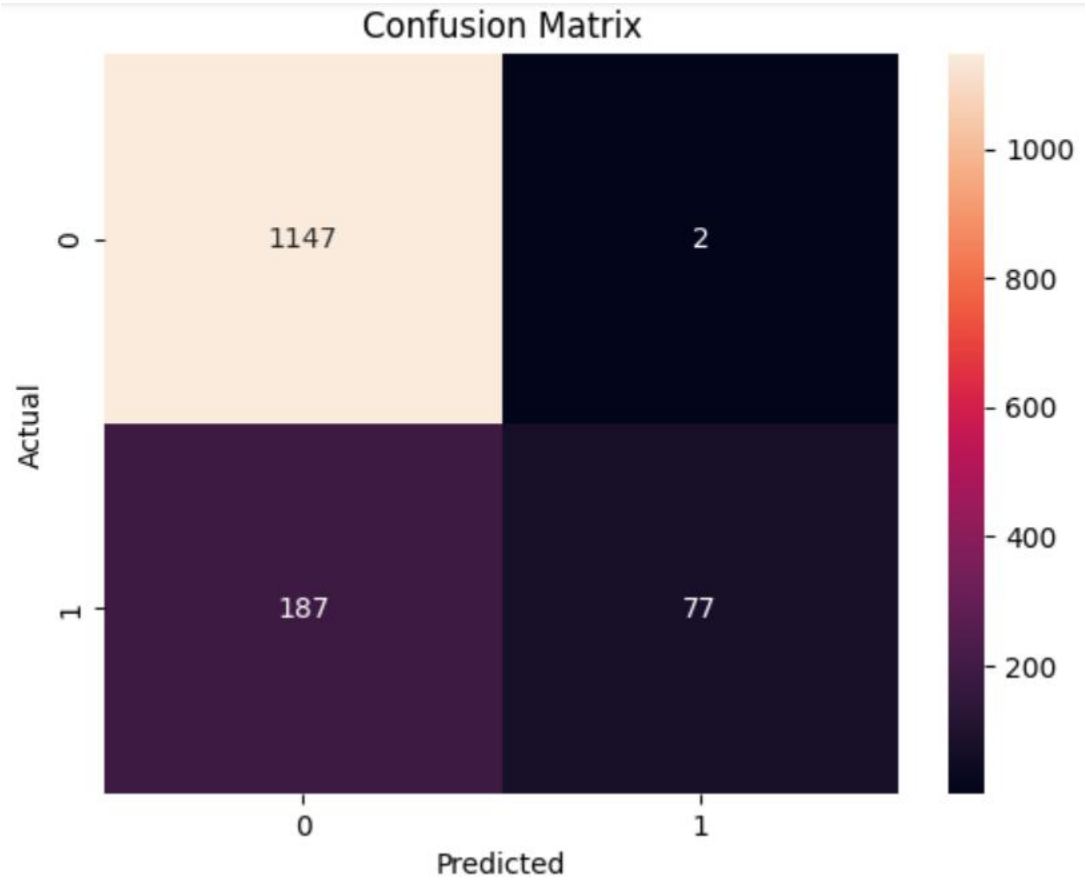
A new instance of DecisionTreeClassifier is created with specific parameters set. These parameters control the behavior of the decision tree, such as the maximum depth of the tree, minimum number of samples required to split an internal node, maximum number of features considered for the best split, criterion for measuring the quality of a split (entropy in this case), and the minimum number of samples required to be at a leaf node.



Classification Accuracy: 0.8650151668351871
Classification Error: 0.1349848331648129
False Positive Rate: 0.0024875621890547263
Precision: 0.963963963963964
F1 score :- 0.4449064449064449
Mean Squared Error: 0.13498483316481294

D. Random Forest Classifier

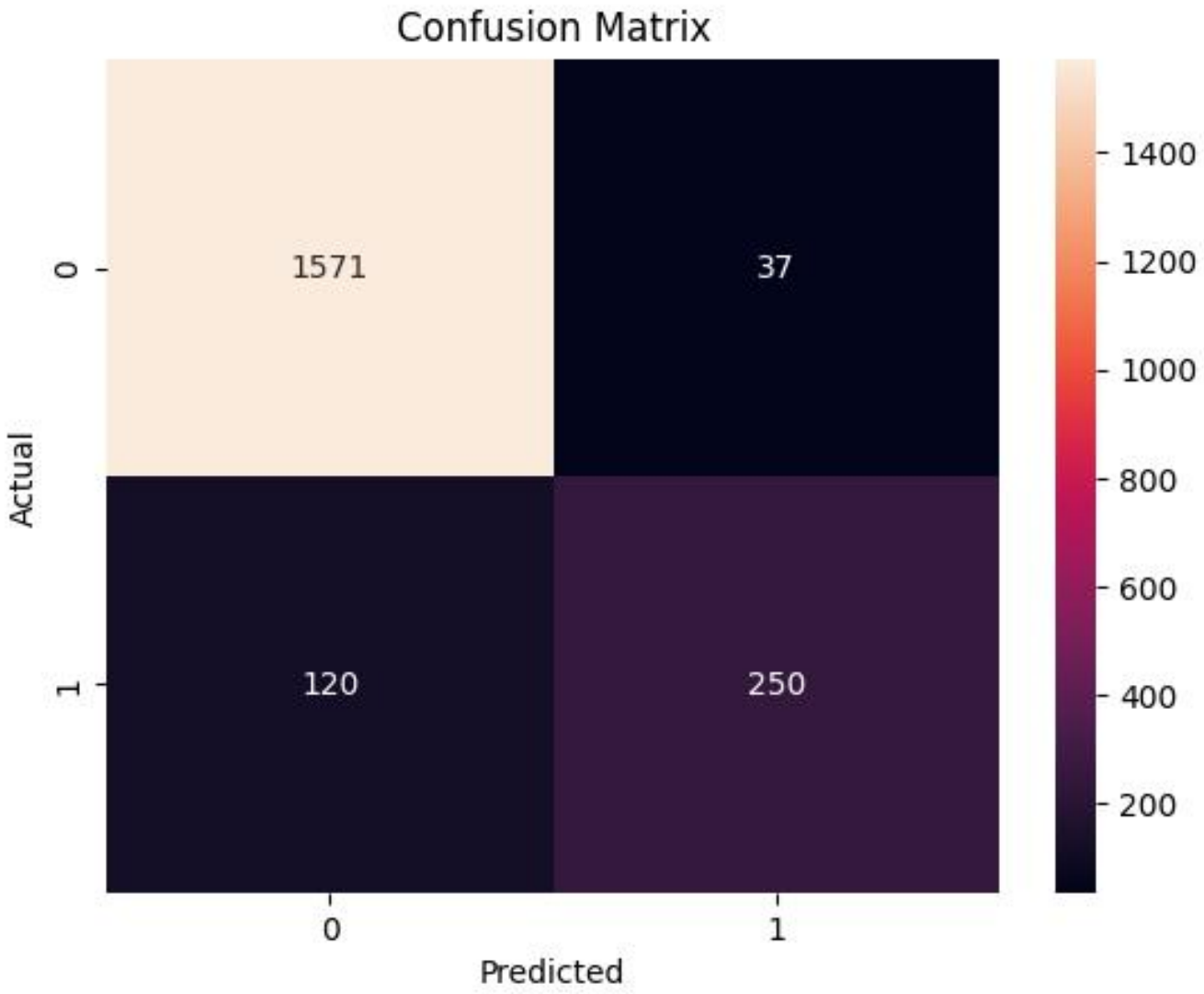
Definition: Random forest is an ensemble machine learning algorithm that combines multiple decision trees to make predictions by averaging or voting the results of individual trees.



Classification Accuracy: 0.9428715874620829
Classification Error: 0.057128412537917 12
False Positive Rate: 0.03233830845771144
Precision: 0.8559556786703602
F1 score :- 0.8454172366621067
Mean Squared Error: 0.057128412537917 09

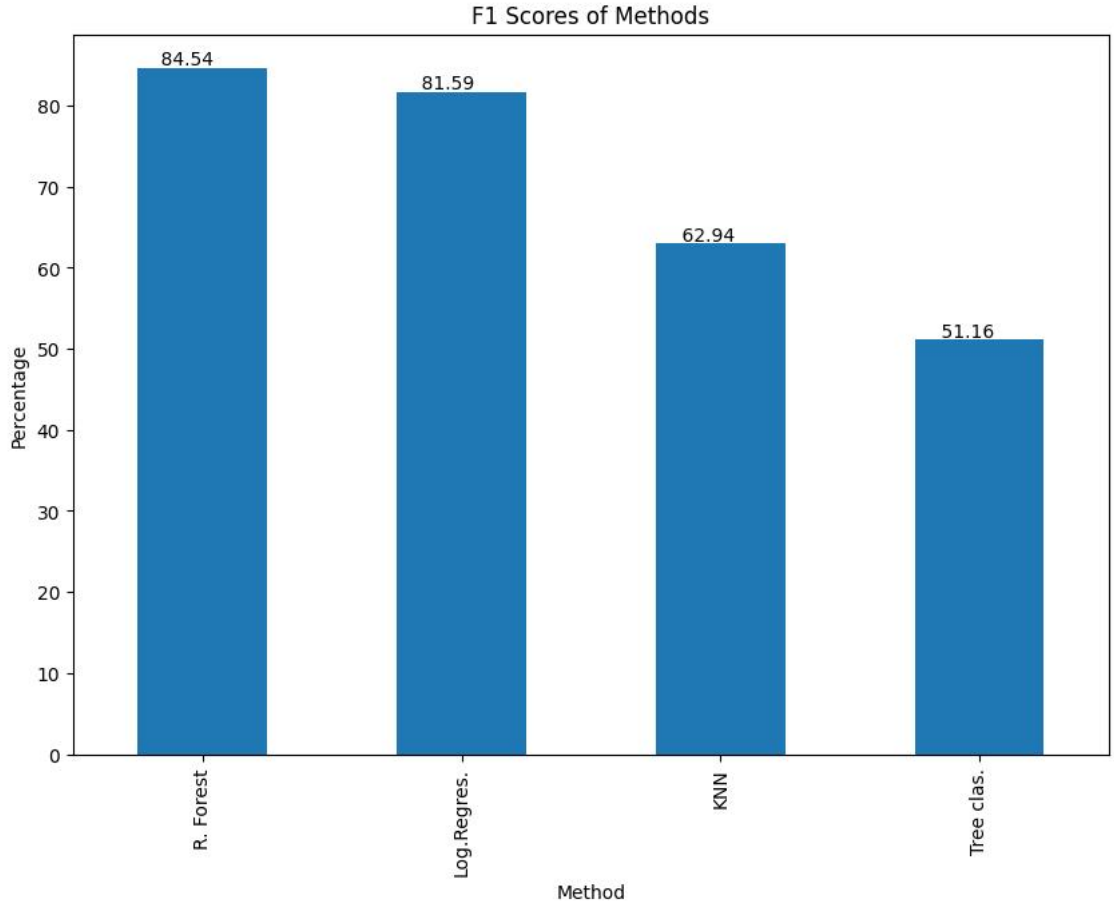
E. Support Vector Machine

Definition: SVM (Support Vector Machine) classifier is a supervised machine learning algorithm that separates data points into different classes using a hyperplane in a high-dimensional space.



8. Plotting the F1 Score for every Algorithm

The code defines a function called `plotSuccess` that takes the `methodDict` dictionary, sorts the success percentages in descending order, and visualizes them as a bar chart. The plot shows the success of different methods, with each bar representing a method and its height representing the success percentage. The plot is annotated with the corresponding success percentages, and labels are provided for the x-axis, y-axis, and title. The function then displays the plot.



9. Checking for Overfitting in Logistic Regression

A. Checking the Mean Absolute error

The provided code is used to calculate the Mean Absolute Error (MAE) for the training and test sets after fitting a Logistic Regression model.

In summary, this code snippet fits a Logistic Regression model, makes predictions on the training and test sets, and calculates the Mean Absolute Error (MAE) between the actual and predicted target values for both sets. The MAE values are then stored in the `mae_train` and `mae_test` lists, respectively.

B. K-fold Cross Validation

By using k-fold cross-validation, this code allows for a more reliable evaluation of the logistic regression model's performance by considering multiple train-test splits and averaging the results.

10. Testing OR Verification Plan

Predicting result on given data

We will be using a Random Forest Classifier as it is providing highest value of F1 Score.

	Job_Description	Actual_label	Predicted_label
0	Architecte d'IA : Conçoit et met en place l'in...	1	1
1	Chercheur en IA : Mène de recherches avancées ...	1	1
2	Éthicien (ne) de l'IA : Évalue et analyse le...	1	1
3	Vendeur/Vendeuse : Responsable de l'accueil et...	0	0
4	Secrétaire : Chargé (e) de l'organisation et...	0	0
5	Ingénieur (e) en informatique : Responsable ...	0	0
6	Serveur/Serveuse : Responsable du service et d...	0	0
7	Mécanicien (ne) automobile : Spécialiste de ...	0	0
8	Enseignant (e) : Chargé (e) d'instruire et...	0	0

Code

1. Importing Libraries

```
import pandas as pd
import pickle
import numpy as np
import sklearn
import nltk
import seaborn as sns
import matplotlib.pyplot as plt

# Importing libraries to check the different accuracy measures
for ML Algos
from sklearn import metrics
from sklearn.metrics import accuracy_score,
mean_squared_error, precision_recall_curve
from sklearn.model_selection import cross_val_score
from sklearn import preprocessing
from sklearn.preprocessing import
LabelEncoder,MinMaxScaler,binarize

# Importing tools to split data and evaluate model performance
from sklearn.model_selection import train_test_split, KFold,
cross_val_score
from sklearn.metrics import f1_score, precision_score,
recall_score, precision_recall_curve, fbeta_score,
confusion_matrix
from sklearn.metrics import roc_auc_score, roc_curve

# Importing libraries for removing stop words
from nltk.corpus import stopwords
from nltk.stem.lancaster import LancasterStemmer

# Importing libraries for pre-processing
from sklearn import preprocessing
from sklearn.feature_selection import SelectFromModel
```

```
# Importing libraries for creating Bag of Words
from sklearn.feature_extraction.text import CountVectorizer

# Importing ML Algos
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier

# Importing libraries to check for Overfitting of our Model
from sklearn.metrics import mean_absolute_error
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
```

2. Loading and Reading the Dataset

```
df=pd.read_pickle("/content/drive/MyDrive/Colab
Notebooks/dataset.pkl")


df.shape


# Displaying all the Columns of our dataset
list(df.columns.values)
```

3. Data Cleaning

A. Display the Null values percentage against every columns (compare to the total number of records)

```
i=0
l = list(df.isnull().sum() * 100 / len(df))
length = len(l)
while i<length:
  if l[i] != 0.0:
    print(df.columns[i]," - ",(l[i]),"% null")
  i+=1
```

B. Removing Rows with label NAN

```
df.dropna(subset= ['label'], inplace=True)
```

C. Replace None in Description column with empty string

```
df2 = df[['description']] = df[['description']].fillna('')
```

D. Concatenating columns jobTitle and Description as jobDescription

```
df["jobDescription"] = df[['jobTitle', 'description']].agg(' '.join, axis=1)
```

E. Converting jobDescription column into lowercase

```
low = lambda s:s.lower()
df['jobDescription']=df['jobDescription'].apply(low)
```

F. Removing stopwords

```
# Removing the Stopwords using Lambda functions
stop_wordsFr= stopwords.words('french')
df['jobDescription_without_stopwords'] =
df['jobDescription'].apply(lambda x: ' '.join([word for word in
x.split() if word not in (stop_wordsFr)]))
stop_wordsEn = stopwords.words('english')
```

```
df['jobDescription_without_stopwords'] =
df['jobDescription_without_stopwords'].apply(lambda x: '
'.join([word for word in x.split() if word not in (stop_wordsEn)]))
```

G. Removing unecessary columns

```
Df=df.drop(['companyUrl','dateRange','location','logoUrl','jobDe
scription'], axis=1)
```

H. Relabelling the label column

```
df['label']=df['label'].replace([0.0,1.0,2.0],0.0)
df['label']=df['label'].replace([3.0,4.0],1.0)
```

4. Exploratory Data Analysis

```
#Count plot to visualize the 'label' column
sns.countplot(data=df,x='label')
plt.show()

# define Seaborn color palette to use
palette_color = sns.color_palette('bright')

labels = ["Previously Didn't Worked on AI","Previously Worked
on AI"]
# plotting data on chart
plt.pie(df.label.value_counts(),labels=labels,shadow=True,wedg
eprops = {"edgecolor" : "black",
                  'linewidth': 1,
                  'antialiased':                                    True},
colors=palette_color,autopct='%.0f%%')

plt.title("Employe's Previous Experience")
```

```
# displaying chart
plt.show()
```

5. Feature Selection and Feature Engineering

```
#Storing the Columns in 2 different as we have to perform Leme
X=df_label_desc.jobDescription_without_stopwords
Y=df_label_desc.label
```

A. Lemmatization

```
import nltk
nltk.download('wordnet')
nltk.download('punkt')
from nltk.stem import WordNetLemmatizer

# Initialize the lemmatizer object
lemmatizer = WordNetLemmatizer()

# Define a function to lemmatize a sentence
def lemmatize_sentence(sentence):
    tokens = nltk.word_tokenize(sentence)
    lemmatized_tokens = []
    for token in tokens:
        lemmatized_token = lemmatizer.lemmatize(token)
        lemmatized_tokens.append(lemmatized_token)
    lemmatized_sentence = ' '.join(lemmatized_tokens)
    return lemmatized_sentence

# Lemmatize the feature column
df_label_desc['jobDescription_without_stopwords']                =
df_label_desc['jobDescription_without_stopwords'].apply(lemm
atize_sentence)
```

B. Creating bag of words

```
cv = CountVectorizer(stop_words=None,max_features=120)

# Tokanise and vectorising our whole testing data
X_vectors = cv.fit_transform(X)
```

6. Splitting

```
X_train, X_test, Y_train, Y_test = train_test_split( X_vectors, Y, test_size=0.35, random_state=42)

X_train.shape, X_test.shape, Y_train.shape, Y_test.shape
```

7. Modelling

```
# This function will help us to print all the different Accuracy measures
def evalClassModel(model, Y_test, Y_pred_class, plot=False):
    # Classification accuracy: percentage of correct predictions
    # calculate accuracy
    print('Accuracy:', metrics.accuracy_score(Y_test, Y_pred_class))
    print('Null accuracy:\n', Y_test.value_counts(normalize=True)*100)
    # calculate the percentage of ones
    print('Percentage of ones:', Y_test.mean())
    # calculate the percentage of zeros
    print('Percentage of zeros:', 1-Y_test.mean())
    # Comparing the first 20 Actual and Predicted response values
    print('True:', Y_test.values[0:20])
    print('Pred:', Y_pred_class[0:20])
```

```
    # Confusion matrix
    # save confusion matrix and slice into four pieces
    confusion = metrics.confusion_matrix(Y_test, Y_pred_class)
    # [row, column]
    TP = confusion[1, 1]
    TN = confusion[0, 0]
    FP = confusion[0, 1]
    FN = confusion[1, 0]
    # visualize Confusion Matrix
    sns.heatmap(confusion, annot=True, fmt="d")
    plt.title('Confusion Matrix')
    plt.xlabel('Predicted')
    plt.ylabel('Actual')
    plt.show()

    # Metrics computed from a confusion matrix
    # Classification Accuracy: Overall, how often is the classifier
correct?
    accuracy = metrics.accuracy_score(Y_test, Y_pred_class)
    print('Classification Accuracy:', accuracy)
    # Classification Error: Overall, how often is the classifier
incorrect?
    print('Classification Error:', 1 - accuracy)
    # False Positive Rate: When the actual value is negative, how
often is the prediction incorrect?
    false_positive_rate = FP / float(TN + FP)
    print('False Positive Rate:', false_positive_rate)
    # Precision: When a positive value is predicted, how often is
the prediction correct?
    print('Precision:',                metrics.precision_score(Y_test,
Y_pred_class))
    #print('AUC        Score:',        metrics.roc_auc_score(Y_test,
Y_pred_class))
    # calculate cross-validated AUC
    # print('Cross-validated AUC:', cross_val_score(model,X, Y,
cv=10, scoring='roc_auc').mean())
    f1 = f1_score(Y_test, Y_pred_class)
    print("F1 score :- ", f1)
    return f1
```

```
methodDict = {}
```

A. Logistic Regression

```
def logisticsRegression():
    logreg=LogisticRegression()
    logreg.fit(X_train,Y_train)
    Y_pred_class=logreg.predict(X_test)
    print('############ Logistic Regression ################')
    f1_score = evalClassModel(logreg, Y_test, Y_pred_class,
True)
    type(Y_test),type(Y_pred_class)
    # Data for final graph
    methodDict['Log.Regres.']=f1_score*100
    # Evaluate the model performance (e.g., using mean squared
error)
    mse = ((Y_pred_class - Y_test) ** 2).mean()
    print("Mean Squared Error:", mse)
logisticsRegression()
```

B. K-Nearest Neighbour

```
def Knn():
    knn = KNeighborsClassifier(n_neighbors=5)
    k_range = list(range(1, 31))
    weight_options = ['uniform', 'disance']
    knn = KNeighborsClassifier(n_neighbors=27,
weights='uniform')
    knn.fit(X_train, Y_train)
    Y_pred_class = knn.predict(X_test)
    print('############ KNeighborsClassifier ################')
    f1_score = evalClassModel(knn, Y_test, Y_pred_class, True)
    # Data for final graph
    methodDict['KNN'] = f1_score * 100
    # Evaluate the model performance (e.g., using mean squared
error)
    mse = ((Y_pred_class - Y_test) ** 2).mean()
    print("Mean Squared Error:", mse)
```

Knn()

C. Decision Tree Classifier

```
def treeClassifier():
    tree = DecisionTreeClassifier()
    # train a decision tree model on the training set
    tree = DecisionTreeClassifier(
        max_depth=3,     min_samples_split=8,     max_features=6,
criterion='entropy', min_samples_leaf=7)
    tree.fit(X_train, Y_train)
    Y_pred_class = tree.predict(X_test)
    print('############ Tree classifier ################')
    f1_score = evalClassModel(tree, Y_test, Y_pred_class, True)
    # Data for final graph
    methodDict['Tree clas.'] = f1_score * 100
    # Evaluate the model performance (e.g., using mean squared
error)
    mse = ((Y_pred_class - Y_test) ** 2).mean()
    print("Mean Squared Error:", mse)

treeClassifier()
```

D. Random Forest Classifier

```
def randomForest():
    # Building and fitting my_forest
    forest = RandomForestClassifier(
        max_depth=None,                    min_samples_leaf=8,
min_samples_split=2, n_estimators=20, random_state=1)
    my_forest = forest.fit(X_train, Y_train)
    # make class predictions for the testing set
    Y_pred_class = my_forest.predict(X_test)
    print('############ Random Forests ################')
    f1_score = evalClassModel(my_forest, Y_test, Y_pred_class,
True)
    # Data for final graph
```

```
    methodDict['R. Forest'] = f1_score * 100
    # Evaluate the model performance (e.g., using mean squared
error)
    mse = ((Y_pred_class - Y_test) ** 2).mean()
    print("Mean Squared Error:", mse)

randomForest()
```

E. Support Vector machine

```
from sklearn import svm

# Create an SVC model
model = svm.SVC()

# Train the model
model.fit(X_train, Y_train)

# Make predictions on the test set
Y_pred_class = model.predict(X_test)

f1_score = evalClassModel(model, Y_test, Y_pred_class, True)
# Data for final graph
methodDict['Support.vector']=f1_score*100
# Evaluate the model performance (e.g., using mean squared
error)
mse = ((Y_pred_class - Y_test) ** 2).mean()
print("Mean Squared Error:", mse)

# Evaluate the model
accuracy = accuracy_score(Y_test, Y_pred_class)
print("Accuracy:", accuracy)
```

8. Plotting the Classification F1 Score for every Algo

```
def plotSuccess():
    s = pd.Series(methodDict)
    s = s.sort_values(ascending=False)
    plt.figure(figsize=(10, 7))
    ax = s.plot(kind='bar')
    for p in ax.patches:
        ax.annotate("    "+str(round(p.get_height(), 2)),
                    (p.get_x()*1.005, p.get_height()*1.005))
    plt.xlabel('Method')
    plt.ylabel('Percentage')
    plt.title('F1 Scores of Methods')
    plt.show()

plotSuccess()
```

9. Checking for overfitting

A. Checking the Mean Absolute Error

```
mae_train=[]
mae_test=[]
logreg=LogisticRegression()
logreg.fit(X_train,Y_train)
y_train_pred=logreg.predict(X_train)
y_test_pred=logreg.predict(X_test)
mae_train.append(mean_absolute_error(Y_train, y_train_pred))
mae_test.append(mean_absolute_error(Y_test, y_test_pred))
```

B. K-fold cross-validation

```
# Assuming you have your input features 'X' and target variable 'Y' defined

# Create an instance of the model you want to evaluate
model = RandomForestClassifier(n_estimators=20)

# Define the number of folds for cross-validation
num_folds = 5

# Create a KFold object
kf = KFold(n_splits=num_folds)

# Perform k-fold cross-validation
scores = cross_val_score(model, X_train, Y_train, cv=kf, scoring='r2')

# Print the cross-validation scores
print("Cross-Validation Scores:", scores)

# Calculate and print the average score
avg_score = scores.mean()
print("Average Score:", avg_score)
```

10. Predicting result on the given data

```
data=pd.read_excel("/content/drive/MyDrive/Checking_Data_AI_LinkedIn.xls")


data['Job_Description'] = data['Job_Description'].apply(lemmatize_sentence)
data.head()


#Creating bag of Words
cv = CountVectorizer(stop_words=None)
X_check_vectors=cv.fit_transform(X_check)
```

```
X_check_vectors.shape


# Building and fitting my_forest
model = RandomForestClassifier(max_depth=None, min_samples_leaf=8, min_samples_split=2, n_estimators=20, random_state=1)
my_forest = model.fit(X_train, Y_train)
predicts=model.predict(X_check_vectors)


data['Predicted_label']=predicts.astype('int')
```

	Job_Description	Actual_label	Predicted_label
0	Architecte d'IA : Conçoit et met en place l'in...	1	1
1	Chercheur en IA : Mène de recherches avancées ...	1	1
2	Éthicien (ne) de l'IA : Évalue et analyse le...	1	1
3	Vendeur/Vendeuse : Responsable de l'accueil et...	0	0
4	Secrétaire : Chargé (e) de l'organisation et...	0	0
5	Ingénieur (e) en informatique : Responsable ...	0	0
6	Serveur/Serveuse : Responsable du service et d...	0	0
7	Mécanicien (ne) automobile : Spécialiste de ...	0	0
8	Enseignant (e) : Chargé (e) d'instruire et...	0	

Conclusion

In conclusion, Artificial Intelligence (AI) has the potential to greatly enhance the process of classifying LinkedIn experiences. By utilizing AI algorithms, it becomes possible to automate the categorization and analysis of job descriptions, skills, and qualifications listed in users' LinkedIn profiles.

AI algorithms can efficiently scan and extract relevant information from LinkedIn experiences, such as job titles, companies, dates, and responsibilities. These algorithms can then use natural language processing and machine learning techniques to classify and categorize the experiences into specific domains, industries, or skill sets.

The use of AI in classifying LinkedIn experiences offers several benefits. It saves time and effort for both job seekers and recruiters by automatically organizing and structuring the vast amount of information available on the platform. This enables faster and more accurate matching of job opportunities with candidates' qualifications.

Additionally, AI algorithms can provide valuable insights and analytics by identifying trends, patterns, and correlations among different experiences. This information can help companies and recruiters gain a better understanding of the skills and expertise available in the job market, enabling them to make more informed decisions.

In summary, AI can significantly improve the classification of LinkedIn experiences, streamlining the process of matching candidates with job opportunities and providing valuable insights for recruiters and companies. As AI continues to advance, it holds great potential for revolutionizing the way we analyze and utilize professional profiles for employment purposes.

Future scope

This project holds a lots of improvement in future. If given more time then I would have worked on transformers to make the results of the project more accurate and improve the performance of the AI system. I would have worked upon the BERT (Bidirectional Encoder Representations from Transformers) model. The advantage of using BERT and transformers in this context lies in their ability to understand the contextual relationships within the text, considering the surrounding words and phrases. This allows for more accurate classification, as the model can capture subtle nuances and dependencies that might be missed by traditional machine learning approaches. In summary, the use of transformers, particularly the BERT model, in the classification of LinkedIn experiences enhances the AI system's ability to understand the semantic meaning and context of the text. This leads to improved accuracy and performance in categorizing job domains, industries, or skill sets, ultimately benefiting job seekers and recruiters in their search for the right opportunities and talents.

Limitations

Data Quality: The effectiveness of AI in classifying LinkedIn experiences heavily relies on the quality and diversity of the training data. If the training data is biased, incomplete, or contains errors, the AI model may produce inaccurate or unreliable classifications.

Contextual Understanding: AI models may struggle to fully comprehend the context and nuances of LinkedIn experiences. The meaning of certain keywords or phrases can vary depending on the industry, role, or company culture. AI algorithms may not always be able to accurately capture these subtleties, leading to misclassifications.

Limited Domain Expertise: AI models are typically trained on a wide range of data from various domains. However, they may lack specific domain expertise required to accurately classify LinkedIn experiences in specialized fields or industries. This can result in lower accuracy and relevance for certain job profiles.

Evolving Terminology: LinkedIn experiences often include industry-specific jargon and evolving terminology. AI models may struggle to keep up with the constant changes in language and may misclassify experiences that include new or uncommon terms that were not present in the training data.

Lack of Human Judgment: AI models operate based on patterns and correlations in the data they were trained on. They may not possess the same level of human judgment and intuition when it comes to classifying LinkedIn experiences. Certain experiences may require subjective assessment or deeper understanding, which AI may not be able to provide.

References

1. Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

2. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. In Advances in neural information processing systems (pp. 5998-6008).

1. https://stackoverflow.com/

2. https://huggingface.co/models

3. https://www.dominodatalab.com/blog/transformers-self-attention-to-the-rescue

4. https://www.techtarget.com/searchenterpriseai/definition/lemmatization#:~:text=Lemmatization%20takes%20a%20word%20and,its%20lemma%2C%20%22walk.%22

5. https://machinelearningmastery.com/gentle-introduction-bag-words-model/

6. https://www.geeksforgeeks.org/what-is-exploratory-data-analysis/

7. https://www.geeksforgeeks.org/

8. https://www.youtube.com/channel/UCNU_lfiiWBdtULKOw6X0Dig

Thomas Danguilhen
Full Signature of Supervisor:
…….

Little Beauty Bisoyi
Full signature of the student:
……..