VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering
J Component report

Programme : M.Tech(CSE WITH BA) – Integrated

Course Title : Artificial Intelligence & Knowledge-Based Systems

Course Code : CSSE3088

Slot : C1+C2+TC1+TC2

Title: Fraudulent Job Postings Prediction

Team Members: Vignesh.N | 19MIA1093

Faculty: Dr. L.M. Jenila Livingston

Sign: Vignesh.N
Date: 25/06/23

DECLARATION

I hereby declare that the report titled "**FRAUDULENT JOB POSTING PREDICTION"** submitted by me to VIT Chennai is a record of bona-fide work undertaken by me under the supervision of Prof. Jenila Livingston ,School of Computer Science and Engineering, Vellore Institute of Technology, Chennai.

Signature of Candidate
VIGNESH N

\

CERTIFICATE

Certified that this project report entitled "**FRAUDULENT JOB POSTING PREDICTION"** is a bonafide work of **VIGNESH N(19MIA1093)** and they carried out the Project work under my supervision and guidance for CSE3088 - Artificial Intelligence & Knowledge-Based Systems.

Jenila Livingston
SCOPE, VIT Chennai

ACKNOWLEDGEMENT

I would like to acknowledge that my assignment has been completed and I am ensuring that this was done by me and not copied.

In this accomplishment, We would like to express my special gratitude to all my teachers and most importantly our Professor Mrs. Jenila Livingston of VIT, without their guidance and feedback it is not possible to complete this project.

TABLE OF CONTENTS

ABSTRACT

Employment fraud is becoming more prevalent. According to CNBC, there were twice as many employment frauds in 2018 than there were in 2017. The state of market today has resulted in substantial unemployment. Numerous people have experienced much less job loss and economic stress as a result of the coronavirus.Such a situation offers con artists the ideal opening. Due to a rare incidence, manypeople are becoming victims of scammers who feed on their despair. Most con artists use this technique to obtain personal information from their victims.

Addresses, bank account information, social security numbers, and other personal information are examples. As a student, I have encountered several of these fraudulent emails. The con artists offer their victims incredibly lucrative career opportunities and then demand payment in exchange. Or they demand money fromthe job seeker in exchange for the promise of employment. This is a dangerous problem that can be addressed through Machine Learning techniques and Natural Language Processing (NLP).

INTRODUCTION

For many people, economic hardship and the impact of the coronavirus have drastically reduced work availability and resulted in job loss. Scammers would love totake advantage of a situation like this. Many individuals are falling prey to these con artists who are preying on people's desperation as a result of an extraordinary event.
The majority of fraudsters do this to obtain personal information from the person theyare attempting to defraud. Addresses, bank account numbers, and social security numbers are examples of personal information. Scammers provide customers with a fantastic job offer and then demand money in exchange.
Alternatively, they may needa financial investment from the job seeker in exchange for the promise of a job.
Because of unemployment, there are a lot of job scams these days. A recruiter canfind a qualified candidate through a variety of websites.

Fake recruiters will sometimes post a job on a job platform for the sole purpose of making money. Many job boards suffer from this issue. People later go to a new job portal in quest of legitimate employment, but phoney recruiters also migrate to this portal. As a result, it's critical to distinguish between legitimate and fictitious employment opportunities. Employment fraud is one of the most severe concerns thathas been addressed in the arena of Online Recruitment Frauds in recent years (ORF).
Many organizations these days like to list their job openings online so that job seekersmay find them quickly and simply.
This could, however, be one form of fraud perpetrated by the con artist.

However,thiscould be a form of scam perpetrated by con artists who offer jobseekers work in exchange for money.

This is a dangerous problem that can be solved using machine learning and naturallanguage processing approaches (NLP).

PROBLEM STATEMENT/PROJECT MOTIVE

This project aims to create a classifier that will have the capability to identify fake and real jobs. The final result is evaluated based on two different models. Since thedata provided has numeric and text features, one model will be used on the text data and anotheron numeric data. The final output will be a combination of the two.
The final model will take in any relevant job posting data and produce a final resultdetermining whether the job is real or not.

LITERATURE SURVEY

Online recruiting fraud detection is a relatively new sector in which little research has been done. There are some indirect methods to solve Online recruitment fraud toa limited extent, such as Email Spam filtering, which prevents sending advertising- related emails to users, anti-phishing techniques to detect fake websites, and countermeasures against opinion fraud to detect the posting of deceptive and misleading fake reviews. Review spam detection, email spam detection, and fake news identification have all received a lot of attention in the realm of online fraud detection, according to many studies.

S.NO	TITLE	AUTHOR /JOURNAL	TECHNIQUE	RESULT
1)	A comparative study on fake job postings predictions using data mining techniques.	Sultana Umme Habiba, Md. Khairul Islam, Farzana Tasnim	DEEP NEURAL NETWORK TECHNIQUES: KNN,RANDOM FOREST,DECISIONTREE, SVM	RANDOM FOREST ACHIEVED AN ACCURACY OF 97%.
2)	Fake job recruitment detection	Karri sai suresh reddy, Karri lakshmana reddy	Along with classifiers Two major types of classifiers, such as single classifier and ensemble classifiers are considered for fraudulent job posts detection.	Random Forest classifier outperforms over its Peer classification tool. The proposed approach achieved accuracy 98.27%

				which is much higherthan the existing methods.
3)	Fake Job Recruitment Detection Using Machine Learning Approach.	Shawni Duttaand Prof.Samir Kumar Bandyopadhyay	Multilayer Perceptron ,Ada Boost Classifier, Gradient Boosting Classifier	Accuracy: MLP(96.14 %), AdaBo ost (97.46 %), Gradie nt Boosti ng Classifi er (97.65 %)
4)	Smart Fraud Detection Framework for Job Recruitments. Fake Job Detection Using Machine Learning	Asad Mehboob, M. S. I. Malik Priya, Akshata, Anushka, Prof. S. S. Karve	Proposed SVM SVM, Random Forest	F1-Score 0.98 - Fraud 0.49 – Notfraud
5)				Accuracy: SVM (95%), RF (97%)

DATASET

This dataset contains 18K job descriptions out of which about 800 are fake. The data consists of both textual information and meta-information about the jobs. The dataset can be used to create classification models which can learn the job descriptions whichare fraudulent

Data Types:

```
df.dtypes
```

```
job_id                  int64
title                  object
location               object
department             object
salary_range           object
company_profile        object
description            object
requirements           object
benefits               object
telecommuting           int64
has_company_logo        int64
has_questions           int64
employment_type        object
required_experience    object
required_education     object
industry               object
function               object
fraudulent              int64
dtype: object
```

NULLL VALUES:

Null Values

title	0
location	340
company_profile	3255
requirements	2548
telecommuting	0
has_company_logo	0
has_questions	0
employment_type	3397
required_experience	6858
required_education	7889
industry	4769
function	6261
fraudulent	0

dtype: int64

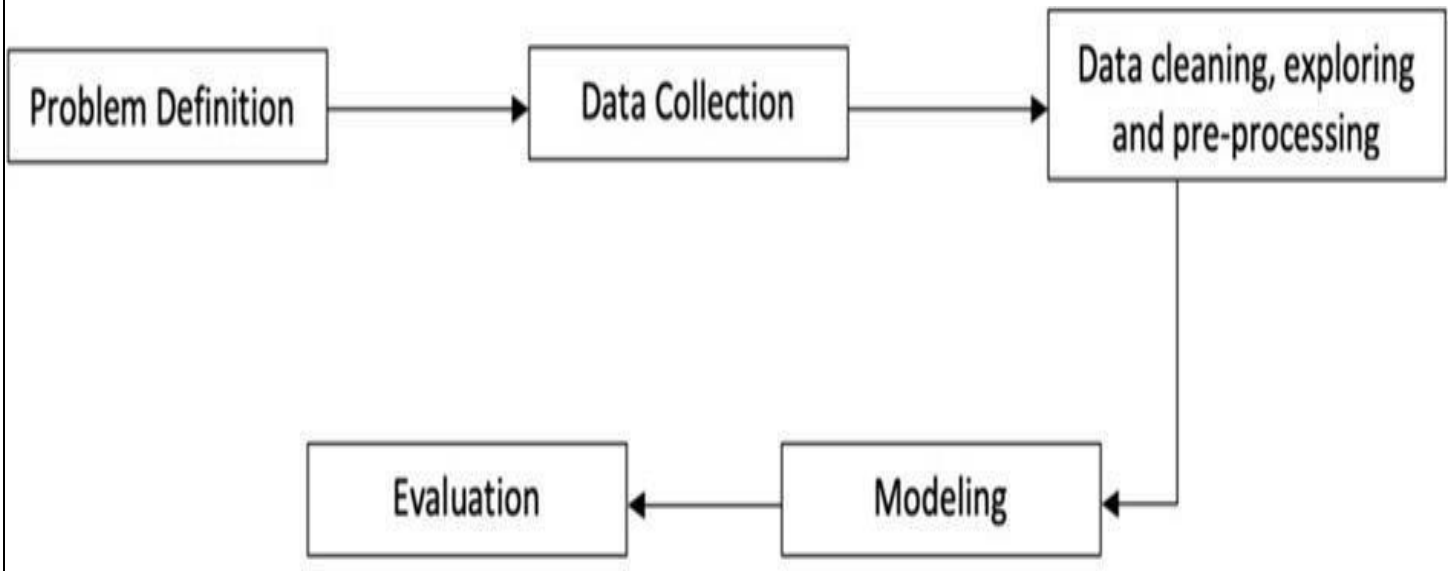
FEATURES:

1. job_id = Unique Job ID
2. title = The title of the job ad entry.
3. location = Geographical location of the job ad.
4. department = Corporate department (e.g. sales).
5. salary_range = Indicative salary range (e.g. $50,000-$60,000)
6. company_profile = A brief company description.
7. description = The detailed description of the job ad.
8. requirements = Enlisted requirements for the job opening.
9. benefits = Enlisted offered benefits by the employer.
10. telecommuting = True for telecommuting positions.
11. has_company_logo = True if company logo is present.
12. has_questions = True if screening questions are present.
13. employment_type = Full-type, Part-time, Contract, etc.
14. required_experience = Executive, Entry level, Intern, etc.
15. required_education = Doctorate, Master’s Degree, Bachelor, etc.
16. industry = Automotive, IT, Health care, Real estate, etc.
17. function = Consulting, Engineering, Research, Sales etc.
18. fraudulent = target - Classification attribute.

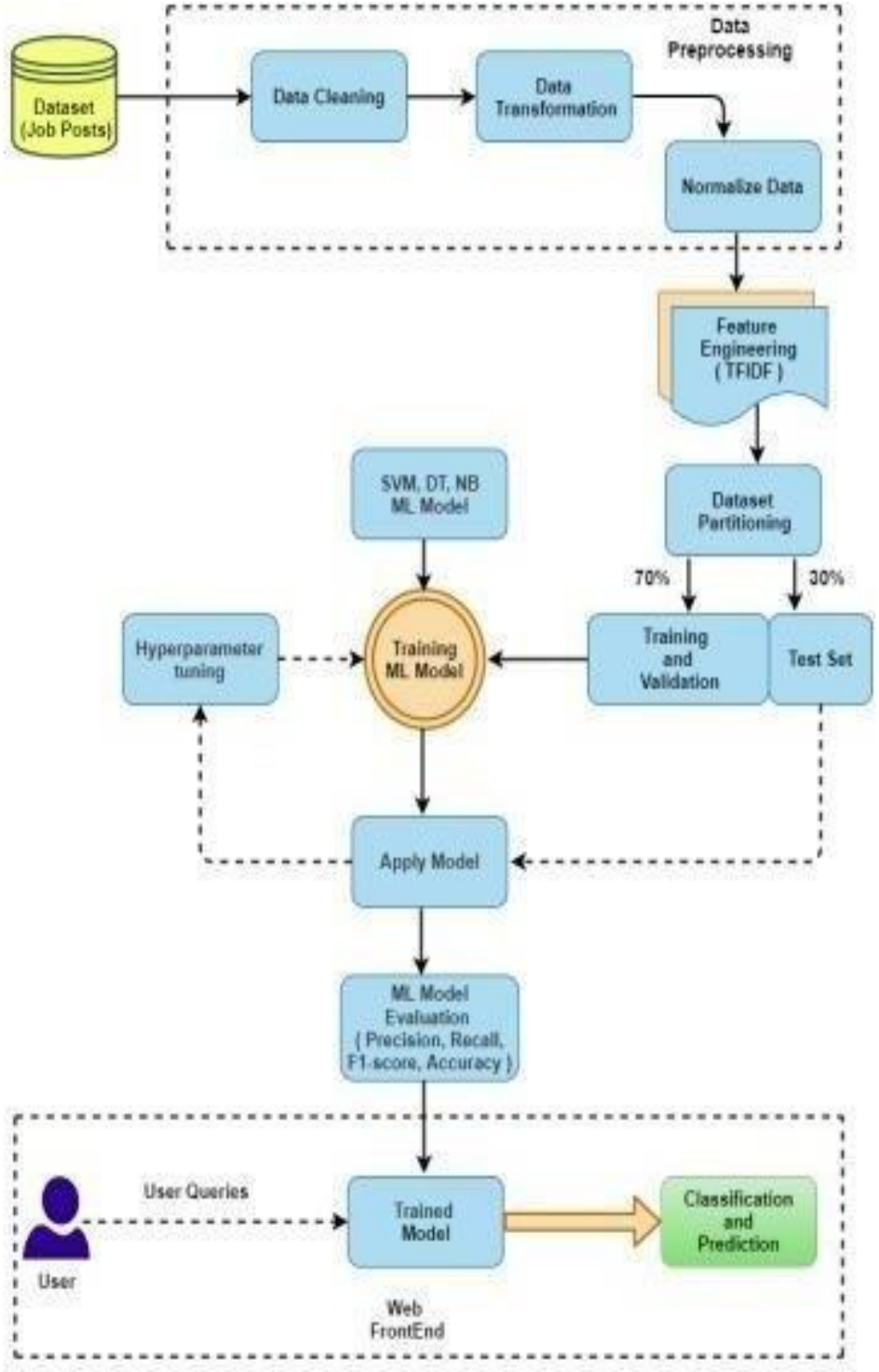
SYSTEM DESIGN

This project follows five stages. The five steps adopted for this project are –

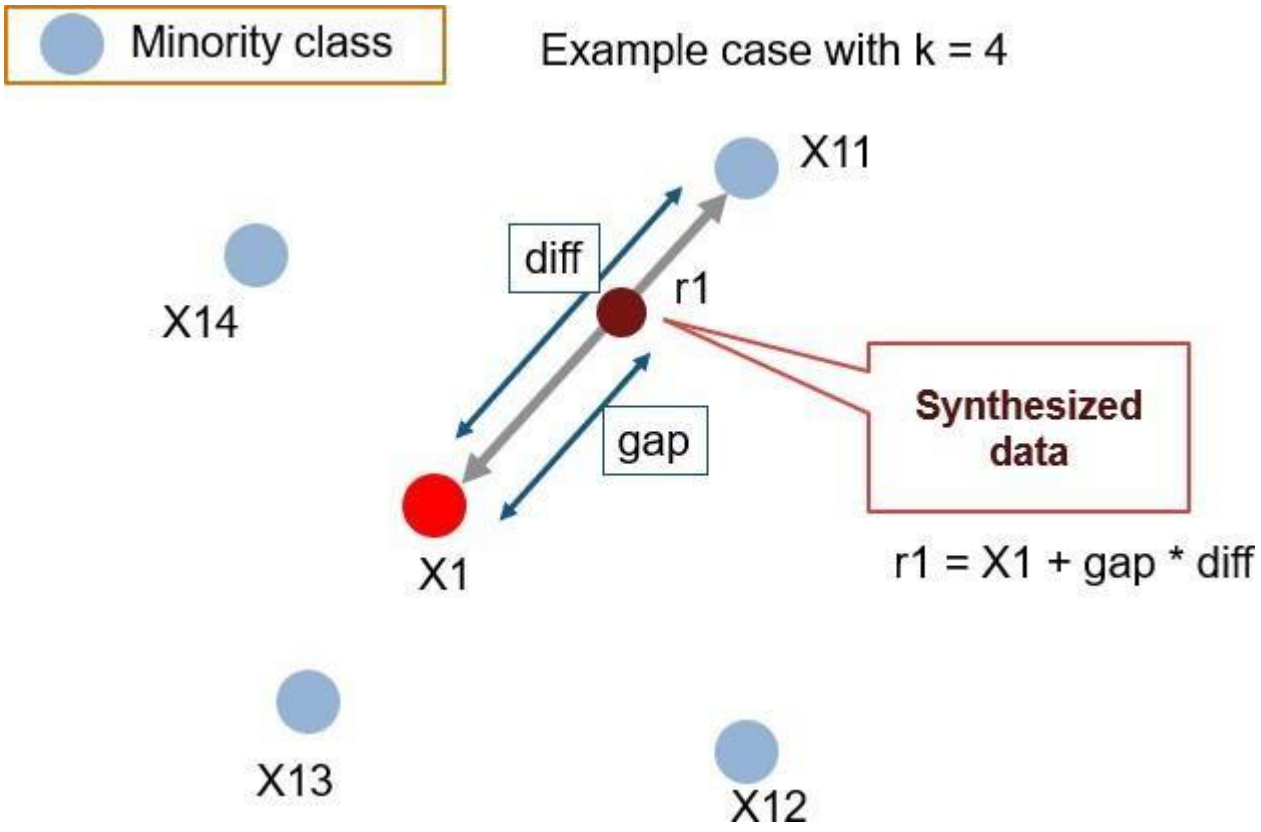
- Problem Definition (Project Overview, Project statement, and Metrics)
- Data Collection
- Data cleaning, exploring, and pre-processing
- Modeling
- Evaluating



OVERALL WORKFLOW DIAGRAM:

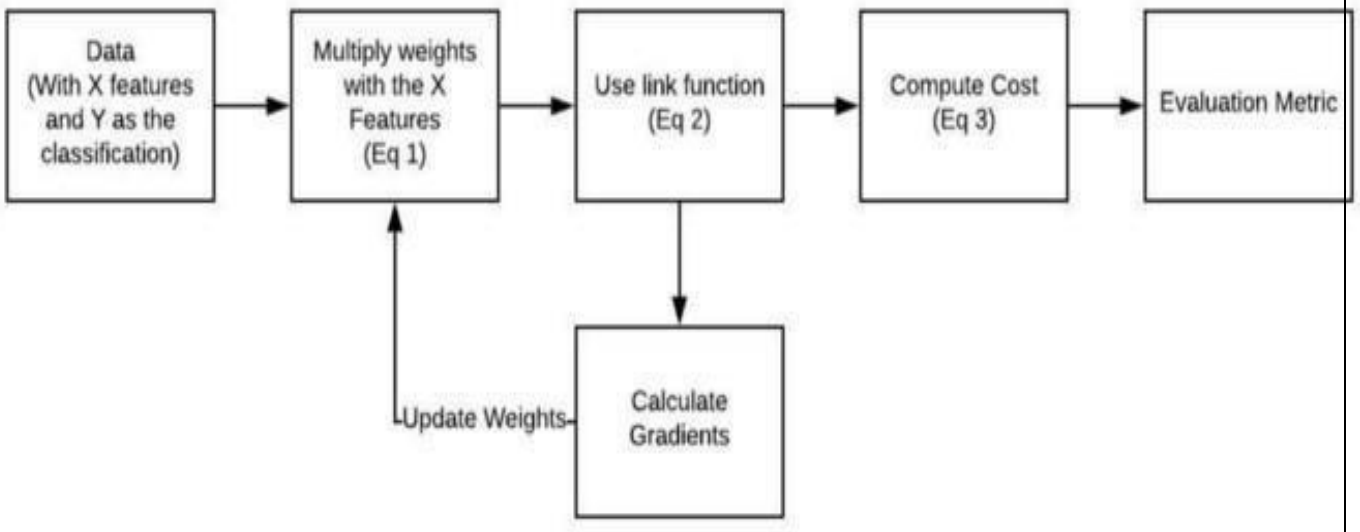


SMOTE ARCHITECTURE DIAGRAM:

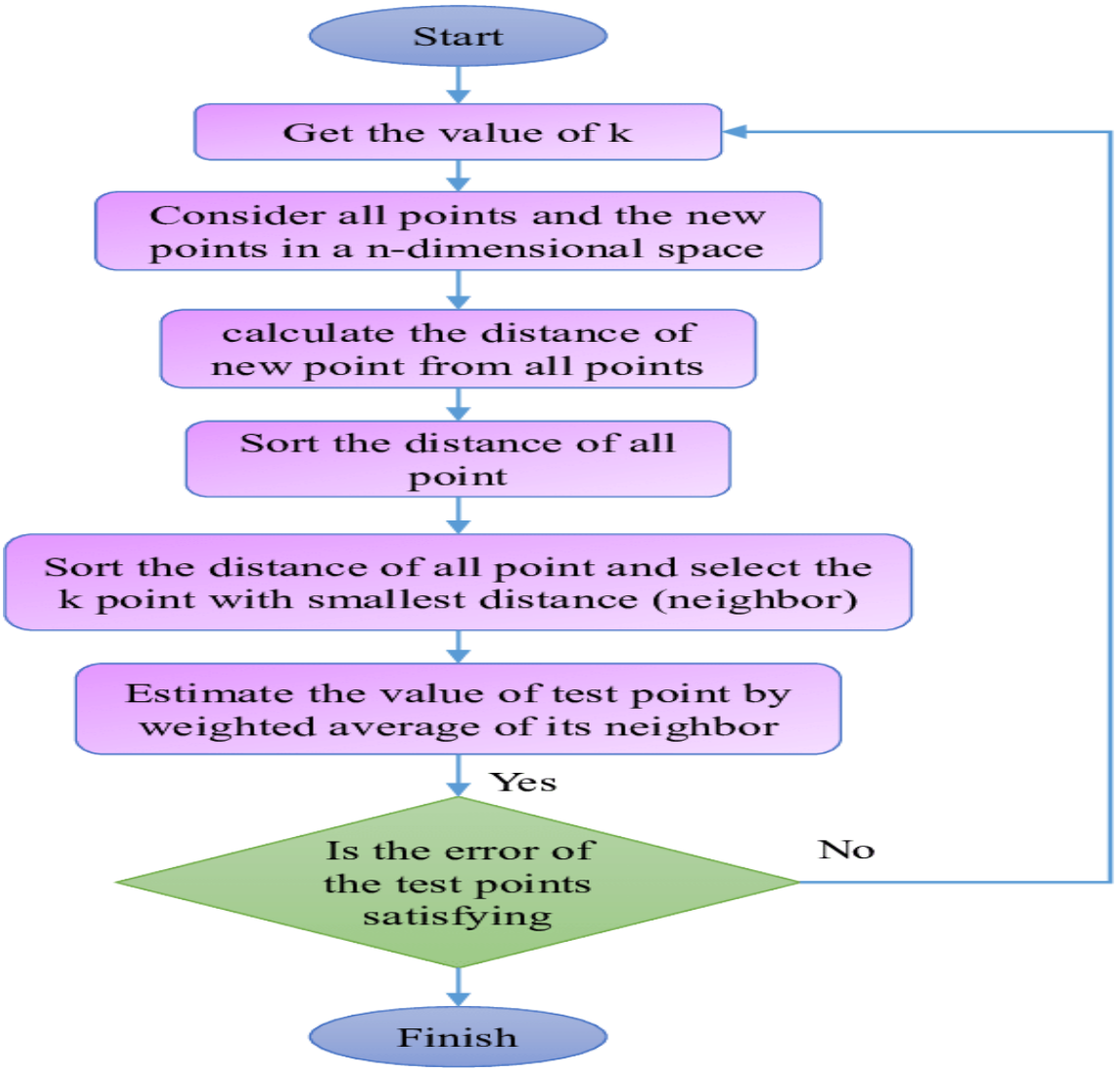


ARCHITECTURE DIAGRAM FOR DIFFERENT MODELS THAT WE HAVEBUILT:

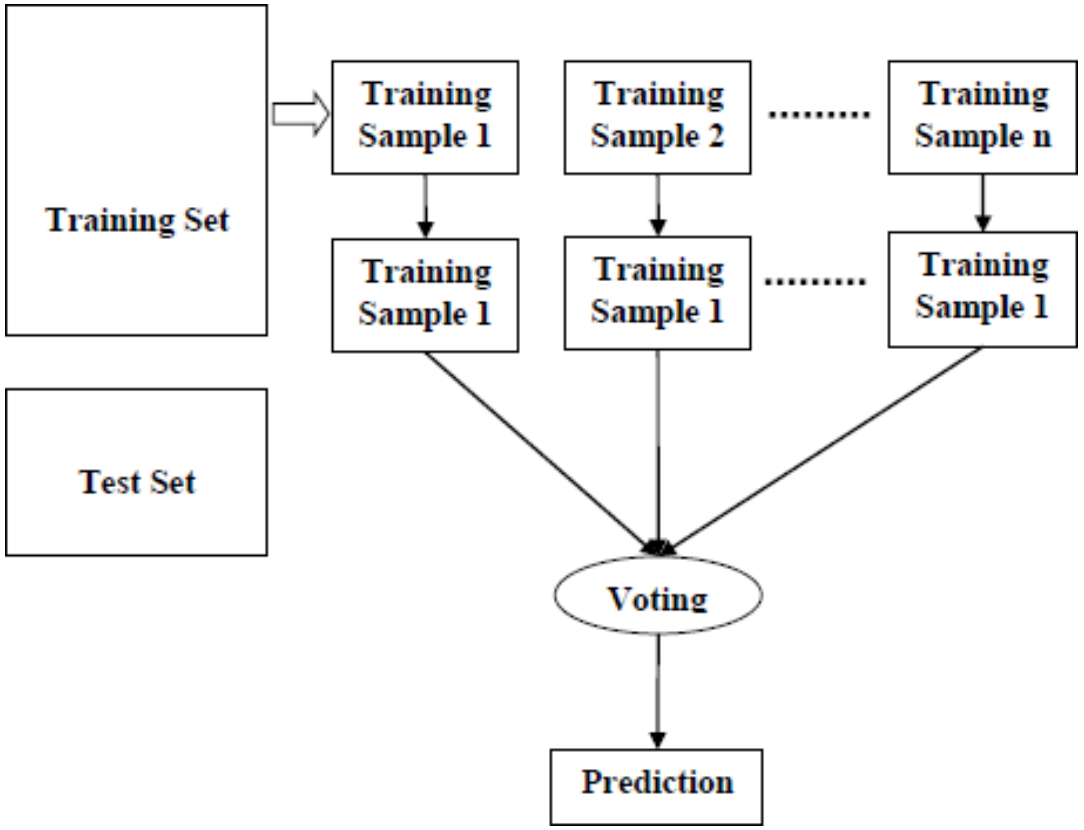
1) LOGISTIC REGRESSION:



2) K NEAREST NEIGHBORS:



3) RANDOM FOREST CLASSIFIER:



IMPLEMENTATION

IMPORTING DATASET:

```
[ ] import numpy as np
    import pandas as pd
    import seaborn as sb
    import matplotlib.pyplot as plt
    import warnings
```

```
[ ] file_path = 'fake job postings.csv'
```

```
[ ] #Reading csv file of dataset
    df = pd.read_csv(file_path)
    df.head(5)
```

EXP

```
[ ] df.columns
```

```
Index(['job_id', 'title', 'location', 'department', 'salary_range',
       'company_profile', 'description', 'requirements', 'benefits',
       'telecommuting', 'has_company_logo', 'has_questions', 'employment_type',
       'required_experience', 'required_education', 'industry', 'function',
       'fraudulent'],
      dtype='object')
```

```
[ ] df.describe()
```

	job_id	telecommuting	has_company_logo	has_questions	fraudulent
count	17880.000000	17880.000000	17880.000000	17880.000000	17880.000000
mean	8940.500000	0.042897	0.795302	0.491723	0.048434
std	5161.655742	0.202631	0.403492	0.499945	0.214688
min	1.000000	0.000000	0.000000	0.000000	0.000000
25%	4470.750000	0.000000	1.000000	0.000000	0.000000
50%	8940.500000	0.000000	1.000000	0.000000	0.000000
75%	13410.250000	0.000000	1.000000	1.000000	0.000000
max	17880.000000	1.000000	1.000000	1.000000	1.000000

Feature Selection:

Out of the 17 features these are the features we have selected for our models

```
[ ] df.columns
```

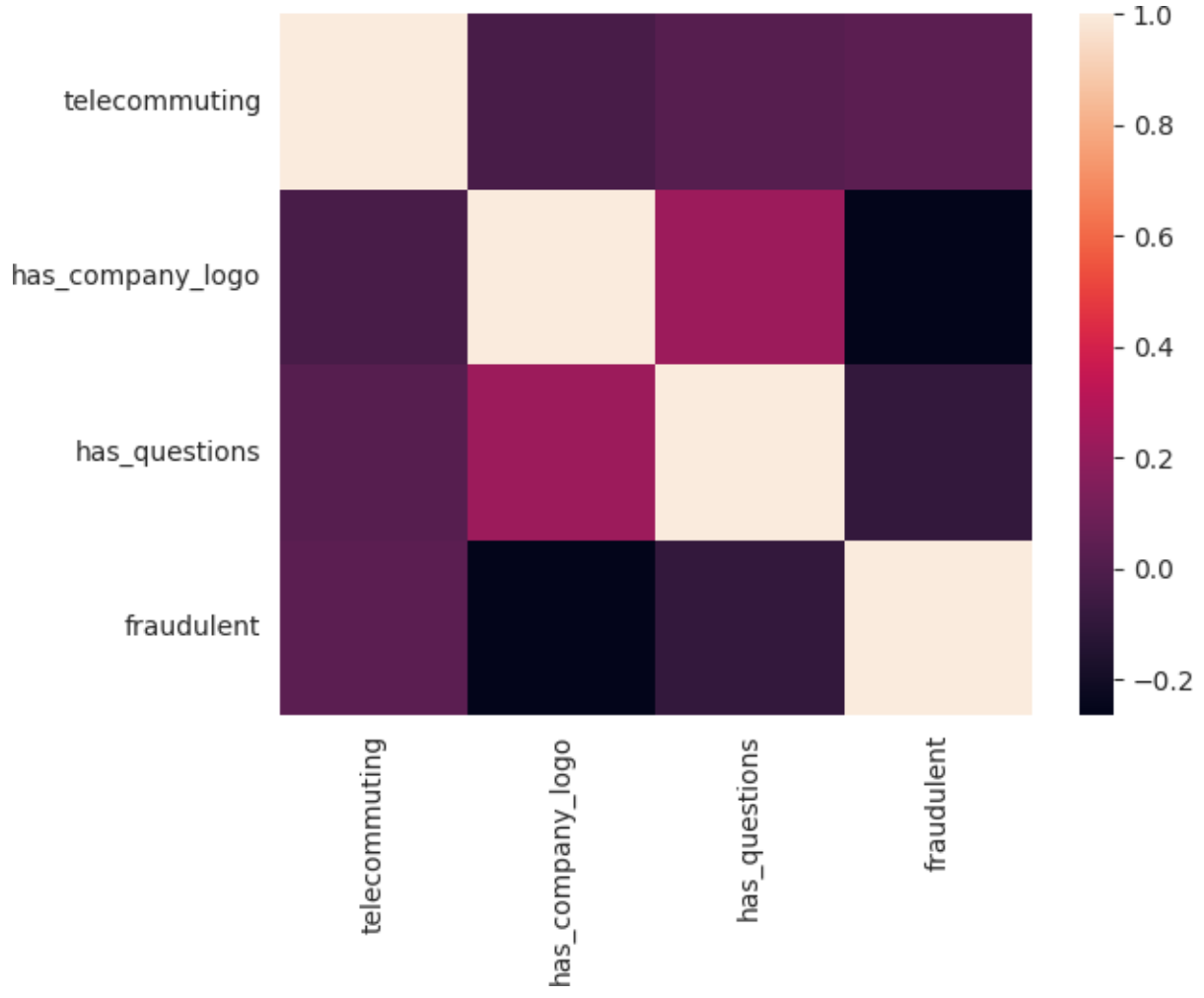
```
Index(['job_id', 'title', 'location', 'department', 'salary_range',
       'company_profile', 'description', 'requirements', 'benefits',
       'telecommuting', 'has_company_logo', 'has_questions', 'employment_type',
       'required_experience', 'required_education', 'industry', 'function',
       'fraudulent'],
      dtype='object')
```

```
[ ] df = df[['title', 'location','company_profile', 'requirements', 'telecommuting', 'has_company_logo', 'has_questions', 'employment_type',
       'required_experience', 'required_education', 'industry', 'function', 'fraudulent']]
```

```
[ ] df.isna().apply(pd.value_counts)
```

	title	location	company_profile	requirements	telecommuting	has_company_logo	has_questions	employment_type	required_experience	required_education
False	17880.0	17534	14572	15185	17880.0	17880.0	17880.0	14409	10830	9775
True	NaN	346	3308	2695	NaN	NaN	NaN	3471	7050	8105

CORRELATION MATRIX:



Next we handle the missing values in the dataset

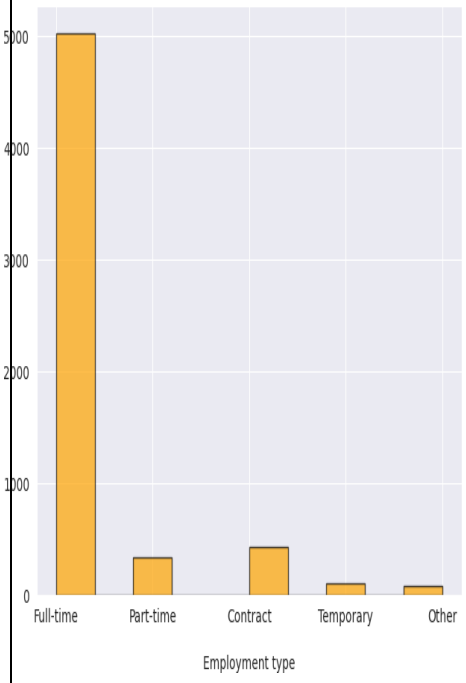
```
[ ]  df.isnull().sum()
```

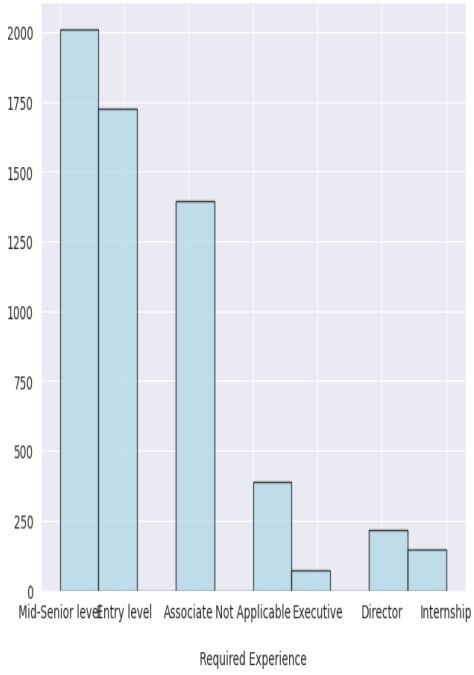
```
title                    0
location               340
company_profile       3255
requirements          2548
telecommuting            0
has_company_logo         0
has_questions            0
employment_type       3397
required_experience   6858
required_education    7889
industry              4769
function              6261
fraudulent               0
dtype: int64
```

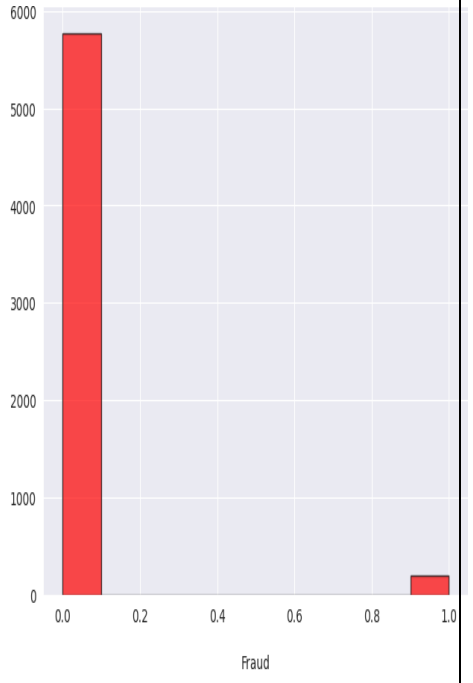
```
[ ]  df.dropna(axis= 0, how= 'any', inplace=True)
```

VISUALIZATION (EDA):

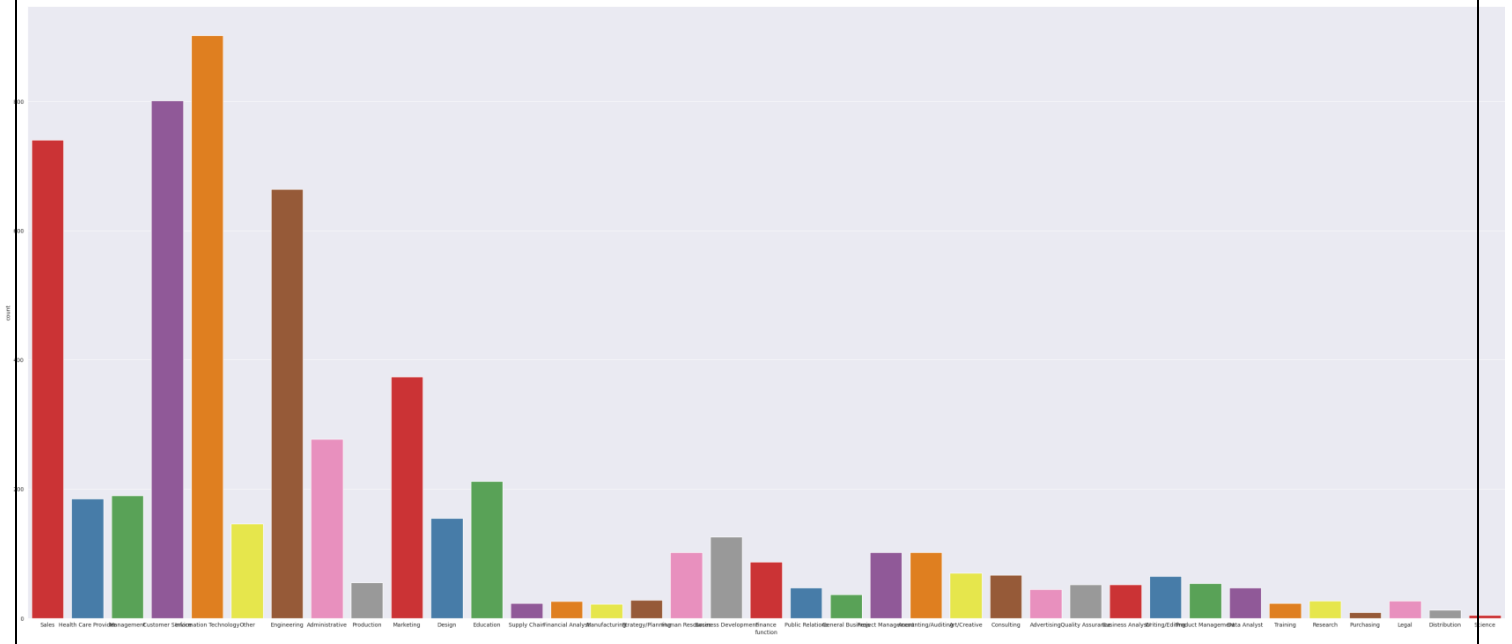
REQUIRED EXPERINCE, EMPLOYEMENT TYPEAND FRAUDULENT DATA







No.of jobs in each area of jobs:



WORD CLOUD USING NLP APPROACH:

```
[ ] df1["text"] = df1["title"]+df1["location"]+df1["description"]+df1["company_profile"]+df1["requirements"]+df1["benefits"]
    cols_to_delete = ["title","location","description","company_profile","requirements","benefits"]
    for col in cols_to_delete:
        del df1[col]
    df1.head()
```

The following steps are taken for text processing:

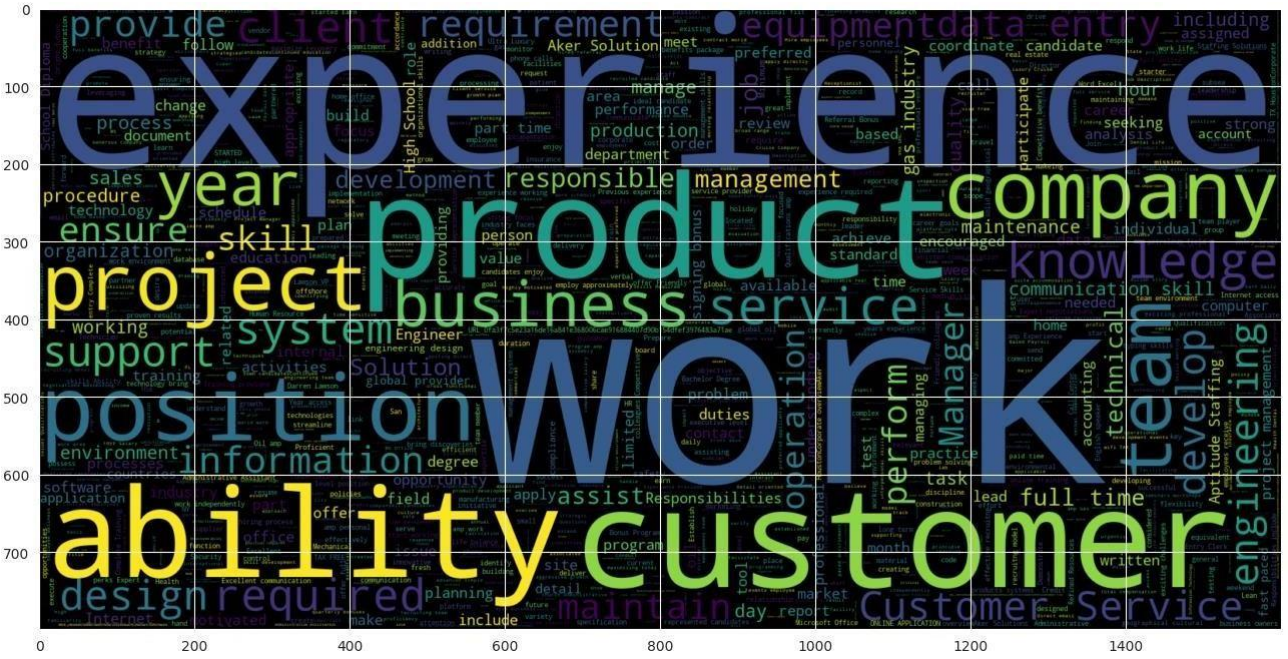
Tokenization: The textual data is split into smaller units. In this case, the data is splitinto words.

To Lower: The split words are converted to lowercase

Stopword removal: Stopwords are words that do not add much meaning to sentences.For example, the, a, an, he, have, etc. These words are removed.

Lemmatization: The process of lemmatization groups in which inflected forms ofwords are used together

WORD CLOUD FOR FRAUDULENT JOB:



WORD CLOUD FOR NON-FRAUDULENT JOBS:

Then we use Label Encoder to convert the labels into a numeric form soas to convert them into the machine-readable form. Machine learning algorithms can then decide in a better way how those labels must be operated.

```
[ ] from sklearn.preprocessing import LabelEncoder
```

```
[ ] le = LabelEncoder()
    df['title'] = le.fit_transform(df['title'])
    df['location'] = le.fit_transform(df['location'])
    df['company_profile'] = le.fit_transform(df['company_profile'])
    df['requirements'] = le.fit_transform(df['requirements'])
    df['employment_type'] = le.fit_transform(df['employment_type'])
    df['required_experience'] = le.fit_transform(df['required_experience'])
    df['required_education'] = le.fit_transform(df['required_education'])
    df['industry'] = le.fit_transform(df['industry'])
    df['function'] = le.fit_transform(df['function'])
```

```
[ ] df = df.reset_index()
    df.head()
```

	index	title	location	company_profile	requirements	telecommuting	has_company_logo	has_questions	employment_type	required_experience	required_education
0	3	115	715	620	1352	0	1	0	1	5	1
1	4	379	730	764	2690	0	1	1	1	5	1
2	6	1618	78	369	4559	0	1	1	1	5	5
3	9	806	554	595	2240	0	1	0	3	2	4
4	12	272	706	595	3245	0	1	0	1	0	1

We perform train, test split and use SMOTE which is an oversampling methods tosolve the imbalance problem.

It aims to balance class distribution by randomly increasing minority class examplesby replicating them.

```
[ ] from sklearn.model_selection import train_test_split
```

```
[ ] X = df[['title', 'location', 'company_profile', 'requirements',
            'telecommuting', 'has_company_logo', 'has_questions', 'employment_type',
            'required_experience', 'required_education', 'industry', 'function']].values
    Y = df[['fraudulent']].values
```

```
[ ] from imblearn.over_sampling import SMOTE
    oversample = SMOTE()
    X, Y = oversample.fit_resample(X, Y)
```

```
[ ] X.shape

    (11536, 12)
```

```
[ ] Y.shape

    (11536,)
```

MODELS:

1. LOGISTIC REGRESSION:

Logistic regression predicts the output of a categorical dependent variable. Therefore, the outcome must be a categorical or discrete value. It can be either Yesor No, 0 or 1, True or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1. Logistic Regression is much similarto Linear Regression except that how they are used. Linear Regression is used for solving.

Regression problems, whereas Logistic regression is used for solving the classification problems.

1) LOGISTIC MODEL

Train the Model

```
[47] from sklearn.linear_model import LogisticRegression

[48] LgR = LogisticRegression()

[49] LgR.fit(X_train, Y_train)
```

```
LogisticRegression()
```

Test the Model

```
[50] Y_pred = LgR.predict(X_test)

[51] Y_test = Y_test.flatten()
     Y_pred = Y_pred.flatten()

[52] Y_test.shape, Y_pred.shape
```

```
((2884,), (2884,))
```

```
[53] df_lgr = pd.DataFrame({'Y_test': Y_test , 'Y_pred': Y_pred})
     df_lgr
```

	Y_test	Y_pred
0	1	1
1	1	1
2	0	0
3	0	1
4	0	0
...	...	...
2879	0	1
2880	0	1
2881	1	1
2882	1	1
2883	0	1

2884 rows × 2 columns

Check Accuracy Score

```
[54] from sklearn.metrics import accuracy_score

[55] accuracy_score(Y_pred, Y_test)
```

```
0.6338418862690708
```

2. K NEAREST NEIGHBORS:

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learningalgorithm that can be used to solve both classification and regression problems. It's easy to implement and understand, but has a major drawback of becoming significantly

slower as the size of that data in use grows.

KNN works by finding the distances between a query and all the examples in the data,selecting the specified number examples (K) closest to the query, then votes for the most frequent label (in the case of classification) or averages the labels (in the case of regression).
In the case of classification and regression, we saw that choosing the right K for ourdata is done by trying several Ks and picking the one that works best.

2) K Nearest Neighbors

Train the Model:

```
[56] from sklearn.neighbors import KNeighborsClassifier

[57] knn = KNeighborsClassifier()

[58] knn.fit(X_train,Y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier()
```

Test the Model

```
[59] Y_pred = knn.predict(X_test)

[60] Y_test = Y_test.flatten()
     Y_pred = Y_pred.flatten()

     df_knn = pd.DataFrame({'Y_test': Y_test , 'Y_pred': Y_pred})
     df_knn
```

	Y_test	Y_pred
0	1	1
1	1	1
2	0	0
3	0	0
4	0	0
...	...	...
2879	0	0
2880	0	0
2881	1	1
2882	1	1
2883	0	0

2884 rows × 2 columns

Checking Accuracy Score

```
[62] accuracy_score(Y_pred,Y_test)
```

```
0.9660194174757282
```

Accuracy using K Nearest Neighbors Algorithm : 96.1% =~ 96%

3. RANDOM FOREST ALGORITHM:

Random Forest is a popular machine learning algorithm that belongs to the supervised learning technique. It can be used for both Classification and Regression problems in ML. It is based on

complex problem and to improve the performance of the model. As the name suggests, "Random Forest is a classifier that contains a number of decision trees on various subsets of the given dataset and takes
the average to improve the predictive accuracy of that dataset."

Instead of relying on one decision tree, the random forest takes the prediction from each tree and based on the majority votes of predictions, and it predicts the final output.

Train the Model

```
[ ] from sklearn.ensemble import RandomForestClassifier
```

```
[ ] rfc = RandomForestClassifier(n_estimators=5)
```

```
[ ] rfc.fit(X_train, Y_train)
```

```
RandomForestClassifier(n_estimators=5)
```

Test the Model:

```
[ ] Y_pred = rfc.predict(X_test)
```

```
[ ] df_rfc = pd.DataFrame({'Y_test': Y_test , 'Y_pred': Y_pred})
    df_rfc
```

	Y_test	Y_pred
0	1	1
1	1	1
2	0	0
3	0	0
4	0	0
...	...	...
2879	0	0
2880	0	0
2881	1	1
2882	1	1
2883	0	0

2884 rows × 2 columns

Check Accuracy Score:

```
[ ] accuracy_score(Y_pred,Y_test)
```

```
0.996879334257975
```

SVM

4) SVM

Support Vector Machines (SVM) is a supervised machine learning algorithm that can be used for both classification and regression tasks. It is particularly effective in handling complex decision boundaries and high-dimensional datasets. SVM aims to find an optimal hyperplane that maximally separates different classes in the feature space.

SVM

```
from sklearn import svm
svm=svm.SVC()
svm.fit(X_train,Y_train)
Y_pred = svm.predict(X_test)
print('Accuracy of SVM Classifier on test set: {:.4f}'.format(svm.score(X_test, Y_test)))
```

```
Accuracy of SVM Classifier on test set: 0.7653
```

```
df_svm = pd.DataFrame({'Y_test': Y_test , 'Y_pred': Y_pred})
df_svm
```

	Y_test	Y_pred
0	1	1
1	1	1
2	0	0
3	0	0
4	0	0
...	...	...
2879	0	1
2880	0	0
2881	1	1
2882	1	0
2883	0	0

2884 rows × 2 columns

5) Decision Tree

A Decision Tree is a supervised machine learning algorithm used for both classification and regression tasks. It creates a model that predicts the value of a target variable based on input features. Decision Trees represent a series of decisions or conditions as a tree-like model, where each internal node represents a test on a feature, each branch represents the outcome of the test, and each leaf node represents the predicted target value.

```
[ ] from sklearn.tree import DecisionTreeClassifier
    dtree=DecisionTreeClassifier()
    dtree.fit(X_train,Y_train)
    Y_pred = dtree.predict(X_test)
```

```
[ ] df_dtree = pd.DataFrame({'Y_test': Y_test , 'Y_pred': Y_pred})
    df_dtree
```

	Y_test	Y_pred
0	1	1
1	1	1
2	0	0
3	0	0
4	0	0
...	...	...
2879	0	0
2880	0	0
2881	1	1
2882	1	1
2883	0	0

2884 rows × 2 columns

```
print("Test Accuracy:", accuracy_score(Y_test, Y_pred))
```

```
Test Accuracy: 0.9947988904299584
```

6) Naive bayes

Naive Bayes is a popular supervised machine learning algorithm used for classification tasks. It is based on Bayes' theorem and assumes that the features are conditionally independent of each other given the class variable. Despite its simplicity, Naive Bayes can be very effective in many real-world scenarios and is particularly useful when working with text classification problems.

```
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

# Create an instance of the GaussianNB classifier
nb = GaussianNB()

# Train the Naive Bayes model
nb.fit(X_train, Y_train)

# Make predictions on the test data
Y_pred = nb.predict(X_test)

# Calculate the accuracy
accuracy = accuracy_score(Y_test, Y_pred)

# Print the accuracy
print("Accuracy:", accuracy)
```

```
Accuracy: 0.7624826629680999
```

CROSS – VALIDATION SCORES

1. Logistic Regression:

```
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
for u in range(5,10):

    kf = KFold(n_splits=u, random_state=None)
    result = cross_val_score(LgR,X,Y, cv = kf)
    print("Avg accuracy when k="+ str(u) +   ":"   "{}".format(result.mean()))
```

```
Avg accuracy when k=5:0.49194118720884894
Avg accuracy when k=6:0.496959104864368
Avg accuracy when k=7:0.5309466019417476
Avg accuracy when k=8:0.5341539528432732
Avg accuracy when k=9:0.5581593131016418
```

2.KNN

```
for u in range(5,15):

    kf = KFold(n_splits=u, random_state=None)
    result = cross_val_score(knn , X, Y, cv = kf)
    print("Avg accuracy when k="+ str(u) +   ":"   "{}".format(result.mean()))
```

```
Avg accuracy when k=5:0.9614217598612917
Avg accuracy when k=6:0.9638456124439913
Avg accuracy when k=7:0.967753120665742
Avg accuracy when k=8:0.9690533980582524
Avg accuracy when k=9:0.9705214646264747
Avg accuracy when k=10:0.9702564780897095
Avg accuracy when k=11:0.971386119029458
Avg accuracy when k=12:0.9714706902532085
Avg accuracy when k=13:0.9719018298499695
Avg accuracy when k=14:0.973127600554785
```

3.Random Forest:

```
for u in range(5,15):

    kf = KFold(n_splits=u, random_state=None)
    result = cross_val_score(rfc , X,Y, cv = kf)
    print("Avg accuracy when k="+ str(u) +   ":"   "{}".format(result.mean()))
```

```
Avg accuracy when k=5:0.9948851322063286
Avg accuracy when k=6:0.9975724245757899
Avg accuracy when k=7:0.9981796116504855
Avg accuracy when k=8:0.9975728155339806
Avg accuracy when k=9:0.9971390736972181
Avg accuracy when k=10:0.9982656200913599
Avg accuracy when k=11:0.9977458621905533
Avg accuracy when k=12:0.997571973638571
Avg accuracy when k=13:0.9972248720839476
Avg accuracy when k=14:0.9985263522884882
```

4.SVM

```
for u in range(5,10):

    kf = KFold(n_splits=u, random_state=None)
    result = cross_val_score(svm,X,Y, cv = kf)
    print("Avg accuracy when k="+ str(u) +   ":"   "{}".format(result.mean()))
```

```
Avg accuracy when k=5:0.5545261989919912
Avg accuracy when k=6:0.5232274604893318
Avg accuracy when k=7:0.5802704576976421
Avg accuracy when k=8:0.5676144244105409
Avg accuracy when k=9:0.6149290624240114
```

5.Decision Tree:

```
[ ] for u in range(5,10):

        kf = KFold(n_splits=u, random_state=None)
        result = cross_val_score(dtree,X,Y, cv = kf)
        print("Avg accuracy when k="+ str(u) +   ":"   "{}".format(result.mean()))
```

```
Avg accuracy when k=5:0.9925444299956654
Avg accuracy when k=6:0.9934111759198082
Avg accuracy when k=7:0.9933252427184466
Avg accuracy when k=8:0.9931518723994452
Avg accuracy when k=9:0.9931496765465206
```

6.Naive bayes

```
[ ] from sklearn.naive_bayes import GaussianNB
    from sklearn.model_selection import KFold, cross_val_score

    # Create an instance of the GaussianNB classifier
    nb = GaussianNB()

    # Define the range of values for k
    k_values = range(5, 15)

    for k in k_values:
        kf = KFold(n_splits=k, random_state=None)
        result = cross_val_score(nb, X, Y, cv=kf)
        print("Avg accuracy when k =", k, ":", result.mean())
```

```
Avg accuracy when k = 5 : 0.6708512784915774
Avg accuracy when k = 6 : 0.6799365314883147
Avg accuracy when k = 7 : 0.7083044382801665
Avg accuracy when k = 8 : 0.7091712898751734
Avg accuracy when k = 9 : 0.7237918920868206
Avg accuracy when k = 10 : 0.7251648551514321
Avg accuracy when k = 11 : 0.7334040251149603
Avg accuracy when k = 12 : 0.732707793842029
Avg accuracy when k = 13 : 0.7347813297644188
Avg accuracy when k = 14 : 0.7355235783633842
```

Methodology with modules

1. Exploring the dataset
2. Data cleaning, exploring, and pre-processing
3. Modeling
4. Evaluation
5. Prediction

Description of Modules

- Data Collection: This module focuses on gathering relevant data for fraudulent job posting prediction. It may involve scraping job postings from online platforms, or obtaining data from other reliable sources. The collected data should include features that can help distinguish between legitimate and fraudulent postings.
- Data Preprocessing: In this module, the collected data is cleaned and prepared for analysis. It includes tasks such as removing duplicates, handling missing values, and addressing inconsistencies in the data. Data preprocessing may also involve standardizing or normalizing numerical features and encoding categorical variables.

- Feature Extraction: This module involves extracting informative features from the job posting data. It may include techniques such as natural language processing (NLP) to extract textual information like job titles, descriptions, and requirements. Feature extraction can also involve extracting numerical features like location, salary, or required experience from structured data.
- Model Selection: In this module, different machine learning algorithms are evaluated and compared to identify the most suitable one for the problem at hand. Various models like logistic regression, decision trees, random forests, support vector machines, or neural networks can be considered. The selection is typically based on the dataset

- Model Training: Once the model is selected, this module involves training the chosen machine learning algorithm on the preprocessed and engineered data. The dataset is divided into training and validation sets to estimate the model's performance. The model is trained by adjusting its parameters or hyperparameters using optimization techniques such as gradient descent or grid search.
- Model Evaluation: This module assesses the trained model's performance using appropriate evaluation metrics. Common metrics for fraud prediction include accuracy, precision, recall, F1 score, and area under the ROC curve. The evaluation helps understand how well the model generalizes to unseen data and detects fraudulent job postings.

CONCLUSION

This system's main purpose was to identify whether a job posting is genuine or not.Job seekers will be able to focus entirely on legitimate job openings if fake job postings are identified and deleted.

We had done data preprocessing, which involves removing things like trivial spaces, null entries, stopwords, and so on. The data was provided to the classifierfor predictions after it had been preprocessed and cleaned to make prediction ready.As we can see below:

Accuracy using Random Forest Classification Algorithm : 99.7%

As per accuracy scores, Random forest algorithm has highest accuracy score, that'swhy for our dataset 'Random Forest' Model is best and suitable to use.

Random Forest for K=13 has Max Accuracy compared to other Models

REFERENCES:

1. B. Alghamdi and F. Alharby, ―An Intelligent Model for Online RecruitmentFraud Detection," J. Inf. Secur., vol. 10, no. 03, pp. 155–176, 2019, doi: 10.4236/jis.2019.103009

2. E. G. Dada, J. S. Bassi, H. Chiroma, S. M. Abdulhamid, A. O. Adetunmbi, and

O. E. Ajibuwa, "Machine learning for email spam filtering: review, approachesand open research problems,‖ Heliyon, vol. 5, no. 6, 2019, doi: 10.1016/j.heliyon.2019.e01802.

3. N. Hussain, H. T. Mirza, G. Rasool, I. Hussain, and M. Kaleem, ―Spam review detection techniques: A systematic literature review,‖ Appl. Sci., vol. 9,no. 5, pp. 1–26, 2019, doi: 10.3390/app9050987.