<u>Registrar of Companies (ROC)</u>

Phase 4 Submission Document

Project: AI-Driven Exploration and Prediction of Company Registration Trends with the Registrar of Companies (ROC)

Building a project like a Registrar of Companies (RoC) involves several steps, including feature engineering, model training, and evaluation. Below is a step-by-step guide to help you approach this project.

1. Define the Problem:

Clearly outline the problem you want to solve with your RoC system. Determine the specific activities you want the system to handle, such as registration of new companies, updating company information, or providing company details to the public.

2. Data Collection:

Gather relevant data for your project. This might include company names, addresses, registration numbers, business types, and other relevant details. Ensure the data is accurate and well-structured.

3. Data Preprocessing:

- Data Cleaning: Handle missing values, outliers, and inconsistencies in the data.

- Feature Engineering: Create relevant features from the raw data. For instance, you could extract the state or country from the address, or generate features based on the registration date.

4. Data Splitting:

Divide your dataset into training and testing sets. The training set is used to train the machine learning model, and the testing set is used to evaluate its performance.

5. Model Selection:

Choose an appropriate machine learning algorithm for your task. For tasks involving text data (like company names and descriptions), natural language processing (NLP) techniques and algorithms like LSTM, GRU, or transformer-based models (such as BERT) can be valuable.

6. Model Training:

Train your selected model using the training data. During training, the model learns to map input features to the correct output (e.g., company type, registration status).

7. Model Evaluation:

Evaluate your model's performance using appropriate metrics. For classification tasks, common metrics include accuracy, precision, recall, and F1-

score. Adjust your model or experiment with different algorithms if the performance is not satisfactory.

8. Hyperparameter Tuning:

Optimize the hyperparameters of your model to improve its performance. Techniques like grid search or random search can be employed for this purpose.

9. Deployment:

Once you have a satisfactory model, deploy it as part of your Registrar of Companies system. This could be a web application, API, or any other interface that allows users to interact with the system.

10. Monitoring and Maintenance:

Continuously monitor the system's performance in real-world scenarios. Update the model and system as new data becomes available or as business requirements change.

Additional Tips:

- **Feature Importance**: Understand which features are most important for your model's predictions. This insight can guide further feature engineering efforts.
- **Data Privacy and Security:** Ensure that you handle sensitive data responsibly and in compliance with applicable data protection laws.

- **Scalability:** Design your system to handle large volumes of data and user requests efficiently.

Html:

```
<!DOCTYPE html>
<html lang="en">


<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-
width, initial-scale=1.0">
    <title>Registrar of Companies</title>
</head>
<body>
    <h1>Registrar of Companies</h1>
    <form id="companyForm">
        <label for="name">Company Name:</label>
        <input type="text" id="name" name="name"
required><br><br>
        <label for="registrationDate">Registration
Date:</label>
        <input type="date" id="registrationDate"
name="registrationDate" required><br><br>
```

```
        <label for="location">Location:</label>
        <input type="text" id="location" name="location"
required><br><br>
       <button type="submit">Submit</button>
    </form>
    <div id="output"></div>
    <script>
document.getElementById("companyForm").addEvent
Listener("submit", function(event) {
            event.preventDefault();
            const formData = new FormData(event.target);
            const jsonData = {};
            formData.forEach((value, key) => {
                jsonData[key] = value;
            });
               fetch('/companies', {
                method: 'POST',
                headers: {
                    'Content-Type': 'application/json'
                },
                body: JSON.stringify(jsonData)
            })
            .then(response => response.json())
            .then(data => {
document.getElementById("output").innerText =
data.message;
            });
        });
    </script>
</body>
</html>
```

Python:

```
# This Python 3 environment comes with many helpful
analytics libraries installed
# It is defined by the kaggle/python Docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O
(e.g. pd.read_csv)
```

```
# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))



import datetime as dt
# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session
```

```
/kaggle/input/all-indian-companies-registration-data-1900-2019/registered_companies.csv
```

In [2]:

```
dataset = pd.read_csv("/kaggle/input/all-indian-companies-registration-data-1900-2019/registered_companies.csv")
```

```
print(dataset.columns)

/opt/conda/lib/python3.7/site-
packages/IPython/core/interactiveshell.py:3166:
DtypeWarning: Columns (10) have mixed types.Specify
dtype option on import or set low_memory=False.

  interactivity=interactivity, compiler=compiler,
result=result)

Index(['CORPORATE_IDENTIFICATION_NUMBER'
, 'COMPANY_NAME', 'COMPANY_STATUS',

       'COMPANY_CLASS',
'COMPANY_CATEGORY',
'COMPANY_SUB_CATEGORY',

       'DATE_OF_REGISTRATION',
'REGISTERED_STATE', 'AUTHORIZED_CAP',

       'PAIDUP_CAPITAL', 'INDUSTRIAL_CLASS',

'PRINCIPAL_BUSINESS_ACTIVITY_AS_PER_CIN',
'REGISTERED_OFFICE_ADDRESS',

       'REGISTRAR_OF_COMPANIES',
'EMAIL_ADDR',
'LATEST_YEAR_ANNUAL_RETURN',

       'LATEST_YEAR_FINANCIAL_STATEMENT'],

      dtype='object')

In [3]:
```

```
col_list = ["COMPANY_NAME",
"COMPANY_STATUS","DATE_OF_REGISTRATIO
N", "REGISTERED_STATE", "EMAIL_ADDR",
"PRINCIPAL_BUSINESS_ACTIVITY_AS_PER_CIN
"]
df = pd.read_csv("/kaggle/input/all-indian-companies-
registration-data-1900-2019/registered_companies.csv",
usecols=col_list)
In [4]:
df = df[df['DATE_OF_REGISTRATION'].notna()]
df = df[df['EMAIL_ADDR'].notna()]


states = ['Maharashtra', 'Karnataka', 'Telangana', 'Delhi',
'Andhra Pradesh', 'Delhi', 'Gujarat']
years = [2020, 2019, 2018, 2017, 2015, 2014, 2013,
2012, 2011, 2010]
df['year'] =
pd.DatetimeIndex(df['DATE_OF_REGISTRATION']).
year
df['month'] =
pd.DatetimeIndex(df['DATE_OF_REGISTRATION']).
month
```

```
search = df.loc
[df['REGISTERED_STATE'].isin(states) &
df['year'].isin(years) &
df['COMPANY_STATUS'].str.contains("ACTV",case=
False)]
```

In [5]:

```
## plot graph

group =
search.groupby('PRINCIPAL_BUSINESS_ACTIVITY
_AS_PER_CIN')

print(group.size())

group.size().plot.pie(y='PRINCIPAL_BUSINESS_ACT
IVITY_AS_PER_CIN', figsize=(40,40), fontsize=40)
```

```
PRINCIPAL_BUSINESS_ACTIVITY_AS_PER_CIN

Activities of private households as employers and
undifferentiated production activities of private
households        60

Agriculture & allied
11425

Construction
23378

Education
7636
```

Electricity gas and water supply	5221
Extraterritorial organizations and bodies	15
Financial intermediation	9413
Health and social work	8552
Hotels and restaurants	6087
Manufacturing	57281
Mining and quarrying	1958
Other community social and personal service activities	15744
Public administration and defence compulsory social security	113
Real estate renting and business activities	167907
Transport storage and communications	11535

```
Wholesale and retail trade repair of motor vehicles
motorcycles and personal and household goods
27700
dtype: int64
Out[5]:
<AxesSubplot:ylabel='None'>
```