

|  |  |  |  |
| --- | --- | --- | --- |
| **DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  **Domain Name : Artificial Intelligence**  **Project Title : AI-Driven Exploration and Prediction of Company Registration Trends with Registrar of Companies (RoC)** | | | |
| **1.** | **Name of the Student (s)** |  |
|  | |  |  |  |  | | --- | --- | --- | --- | | **S.No** | **Name of the Student** | **E-Mail ID** | **Phone No.** | | **1** | Mathan Raj.M | mathanrajm2003@gmail.com | 6379152566 | | **2** | Karthikeyan M | karthikn6482@gmail.com | 9363313534 | | **3** | Monish M | monishkhokho11@gmail.com | 9345340951 | | 4 | Vijay Sanjay kumar K.r | krvskevil3@gmail.com | 8015539457 | | 5 | Varun S.P | varunleela4@gmail.com | 9345560205 | | | |
| **2.** | **Name of the Guide** | : Ms.S.Suganya |
|  | **Department/ Designation** | : CSE**/**AP |
|  | **Institutional Address** | : Chettinad College of Engineering  and Technology  NH-67, Karur-Trichy Highway,  Puliyur CF, Karur |
|  | **Phone No. & Mobile No.** | : 6374527207 |

AI-Driven Exploration and Prediction of Company Registration Trends with (RoC)

**Section 1: Import Necessary Libraries**

# Import necessary libraries

import pandas as pd # Pandas for data manipulation

import numpy as np # NumPy for numerical operations

from sklearn.model\_selection import train\_test\_split # Scikit-Learn for data splitting

from sklearn.ensemble import RandomForestClassifier # Random Forest Classifier

from sklearn.preprocessing import OneHotEncoder # One-hot encoding of categorical variables

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix # Evaluation metrics

**Description:**

This section begins by importing essential Python libraries required for data manipulation, machine learning, and performance evaluation.

**Section 2: Load Your Dataset**

# Load your dataset

data = pd.read\_csv('Data\_Gov\_Tamil\_Nadu.csv', encoding='ISO-8859-1')

**Description:**

In this section, the dataset is loaded from 'Data\_Gov\_Tamil\_Nadu.csv' using the 'ISO-8859-1' encoding. This step prepares the dataset for analysis.

**Section 3: Data Preprocessing**

# Handle missing values, duplicates, and outliers

data.dropna(inplace=True)

data.drop\_duplicates(inplace=True)

from scipy.stats import zscore

data = data[(np.abs(zscore(data.select\_dtypes(include=[np.number]))) < 3).all(axis=1)]

**Description:**

In this part, data preprocessing is performed, including:

Removing rows with missing values.

Eliminating duplicate rows to ensure data integrity.

Calculating z-scores for numeric columns and removing rows with z-scores exceeding 3, addressing outliers.

**Section 4: Define Dependent and Independent Variables**

# Define the dependent and independent variables

X = data.drop('COMPANY\_STATUS', axis=1)

y = data['COMPANY\_STATUS']

**Description:**

In this section, the independent variables (X) are defined by dropping the 'COMPANY\_STATUS' column, and the dependent variable (y) is set as 'COMPANY\_STATUS.'

**Section 5: List of Categorical Columns for One-Hot Encoding**

# List of categorical columns for one-hot encoding

categorical\_columns = [

'CORPORATE\_IDENTIFICATION\_NUMBER', 'COMPANY\_NAME', 'COMPANY\_CLASS',

'COMPANY\_CATEGORY', 'COMPANY\_SUB\_CATEGORY', 'DATE\_OF\_REGISTRATION',

'INDUSTRIAL\_CLASS', 'PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN',

'REGISTERED\_OFFICE\_ADDRESS', 'REGISTRAR\_OF\_COMPANIES', 'EMAIL\_ADDR',

'LATEST\_YEAR\_ANNUAL\_RETURN', 'LATEST\_YEAR\_FINANCIAL\_STATEMENT'

]

**Description:**

Here, a list of columns that contain categorical data and need one-hot encoding is specified. This step is crucial for converting categorical data into a format suitable for machine learning.

**Section 6: Create a One-Hot Encoder**

# Create a one-hot encoder

encoder = OneHotEncoder(sparse=False, handle\_unknown='ignore')

**Description:**

An instance of the OneHotEncoder is created with specific parameters, including options to handle unknown categories and generate non-sparse arrays.

**Section 7: Split the Data into Training and Testing Sets**

# Split the data into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

**Description:**

In this part, the dataset is divided into training and testing sets, with 20% reserved for testing. The 'random\_state' parameter ensures reproducibility.

**Section 8: Fit and Transform the Encoder on Categorical Columns**

# Fit and transform the encoder on the categorical columns for both training and testing sets

X\_train\_encoded = encoder.fit\_transform(X\_train[categorical\_columns])

X\_test\_encoded = encoder.transform(X\_test[categorical\_columns])

**Description:**

The one-hot encoder is fitted on the training set to learn category mappings and is then applied to both the training and testing sets.

**Section 9: Convert Encoded Arrays into DataFrames**

# Convert the encoded arrays into DataFrames

X\_train\_encoded\_df = pd.DataFrame(X\_train\_encoded, columns=encoder.get\_feature\_names(categorical\_columns))

X\_test\_encoded\_df = pd.DataFrame(X\_test\_encoded, columns=encoder.get\_feature\_names(categorical\_columns))

**Description:**

In this section, the encoded arrays are converted into DataFrames, making it more convenient to work with the transformed data.

**Section 10: Concatenate Encoded Data with Numeric Data**

# Concatenate the encoded data with the numeric data

X\_train\_final = pd.concat([X\_train\_encoded\_df, X\_train.drop(categorical\_columns, axis=1)], axis=1)

X\_test\_final = pd.concat([X\_test\_encoded\_df, X\_test.drop(categorical\_columns, axis=1)], axis=1)

**Description:**

Here, the encoded categorical data is combined with the remaining numeric data to create the final training and testing datasets.

**Section 11: Create and Train a Random Forest Classifier**

# Create and train a Random Forest Classifier

rf\_classifier = RandomForestClassifier(n\_estimators=100, random\_state=42)

rf\_classifier.fit(X\_train\_final, y\_train)

**Description:**

A Random Forest Classifier is instantiated with 100 trees and a fixed random state. It is then trained on the final training dataset.

**Section 12: Make Predictions on the Test Data**

# Make predictions on the test data

y\_pred = rf\_classifier.predict(X\_test\_final)

**Description:**

The trained model is applied to the test data, generating predictions.

**Section 13: Model Performance Analysis**

# Model Performance Analysis

accuracy = accuracy\_score(y\_test, y\_pred)

precision = precision\_score(y\_test, y\_pred, average='weighted')

recall = recall\_score(y\_test, y\_pred, average='weighted')

f1 = f1\_score(y\_test, y\_pred, average='weighted')

conf\_matrix = confusion\_matrix(y\_test, y\_pred)

**Description:**

This part calculates various performance metrics, including accuracy, precision, recall, and F1-score, as well as generates a confusion matrix to assess the model's effectiveness.

**Section 14: Output Model Performance Metrics**

# Output model performance metrics

print(f"Accuracy: {accuracy}")

print(f"Precision: {precision}")

print(f"Recall: {recall}")

print(f"F1 Score: {f1}")

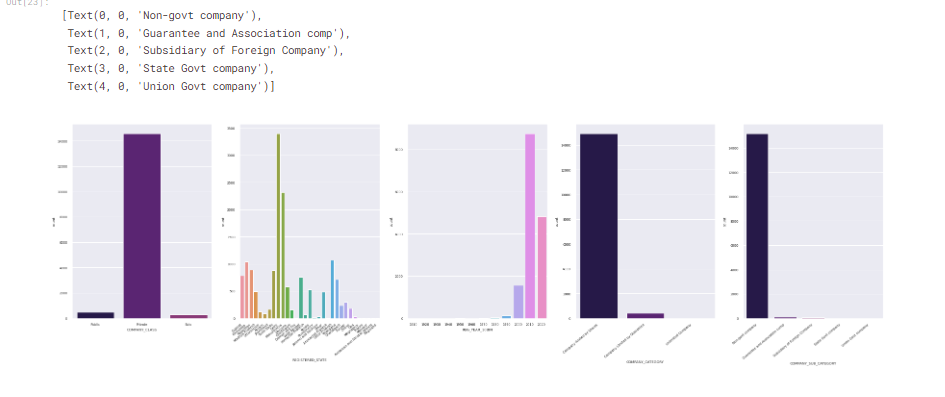
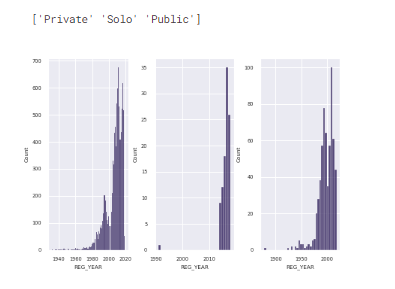
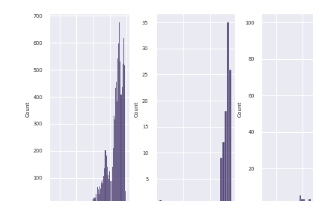
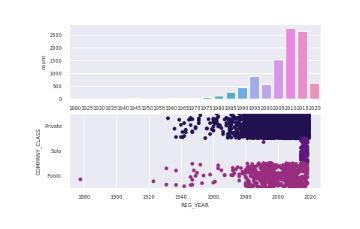
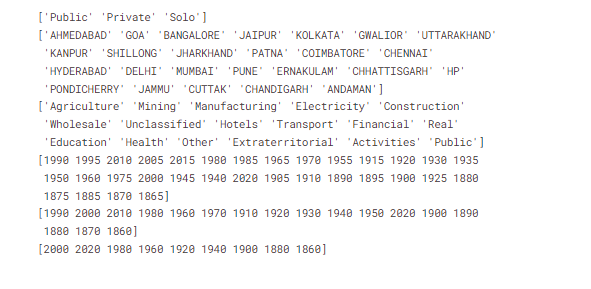
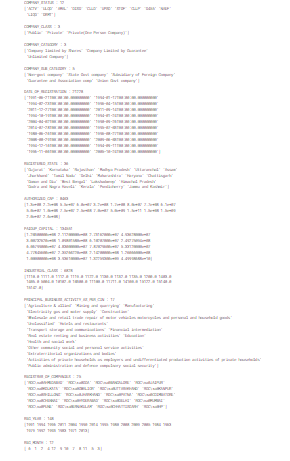
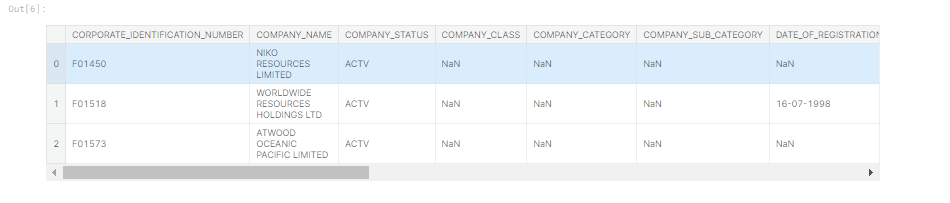
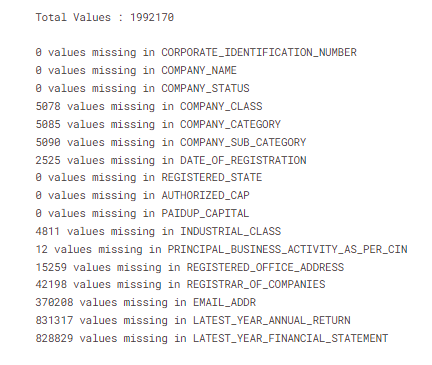
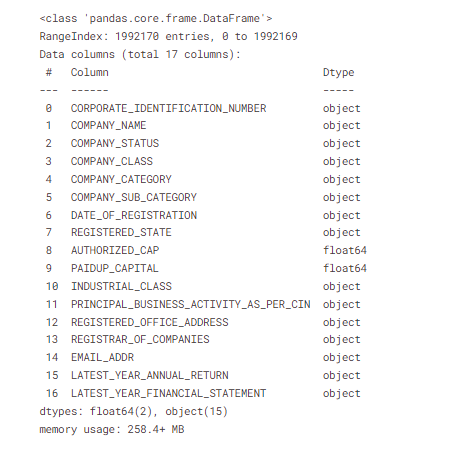
print("Confusion Matrix:")

print(conf\_matrix)

**Description:**

Finally, the program prints the computed performance metrics, allowing the user to assess the model's performance. This includes accuracy, precision, recall, F1 score, and a detailed confusion matrix.

**OutPut :**

****