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

**TEAM LEADER**

**510521104304:EZHUMALAI P**

**PHASE-1:DOCUMENT SUBMISSION**

****

**OBJECTIVIE:**

The problem is to perform an AI-driven exploration and predictive analysis on the master details of companies registered with the Registrar of Companies (RoC). The objective is to uncover hidden patterns, gain insights into the company landscape, and forecast future registration trends.

**PHASE-1: Problem Definition and Design Thinking**

Data Source: Utilize the dataset containing information about registered companies, including columns like company name, status, class, category, registration date, authorized capital, paid-up capital, and more.

**Dataset Link:** [**https://tn.data.gov.in/resource/company-master-data-tamil-nadu-upto-28th-february-2019**](https://tn.data.gov.in/resource/company-master-data-tamil-nadu-upto-28th-february-2019)

**1.Data Source;**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| F00643 | HOCHTIEFF AG, | NAEF | NA | NA | NA | ######## | Tamil Nadu | 0 |
| F00721 | SUMITOMO CORPORATION (SUMITOMO SHOJI KAISHA LIMITED) | ACTV | NA | NA | NA | NA | Tamil Nadu | 0 |
| F00892 | SRILANKAN AIRLINES LIMITED | ACTV | NA | NA | NA | 1/3/1982 | Tamil Nadu | 0 |
| F01208 | CALTEX INDIA LIMITED | NAEF | NA | NA | NA | NA | Tamil Nadu | 0 |
| F01218 | GE HEALTHCARE BIO-SCIENCES LIMITED | ACTV | NA | NA | NA | NA | Tamil Nadu | 0 |
| F01265 | CAIRN ENERGY INDIA PTY. LIMITED | NAEF | NA | NA | NA | NA | Tamil Nadu | 0 |
| F01269 | TORIELLI S.R.L | ACTV | NA | NA | NA | 5/9/1995 | Tamil Nadu | 0 |
| F01311 | HARDY EXPLORATION & PRODUCTION (INDIA) INC.. | ACTV | NA | NA | NA | NA | Tamil Nadu | 0 |
| F01314 | HOCHTIOF AKTIENGESELLSHARFF VORM GFBR HELFMANN | ACTV | NA | NA | NA | ######## | Tamil Nadu | 0 |
| F01412 | EPSON SINGAPORE PVT LTD | ACTV | NA | NA | NA | 25-04-1997 | Tamil Nadu | 0 |
| F01426 | CARGOLUX AIRLINES INTERNATIONAL S A | ACTV | NA | NA | NA | ######## | Tamil Nadu | 0 |
| F01468 | CHO HEUNG ELECTRIC INDUSTRIAL COMPANY LIMITED | NAEF | NA | NA | NA | NA | Tamil Nadu | 0 |
| F01543 | NYCOMED ASIA PACIFIC PTE LIMITED | ACTV | NA | NA | NA | 27-10-1998 | Tamil Nadu | 0 |
| F01544 | CHERRINGTON ASIA LTD | ACTV | NA | NA | NA | 1/5/2000 | Tamil Nadu | 0 |
| F01563 | SHIMADZU ASIA PACIFIC PTE LIMITED | NAEF | NA | NA | NA | NA | Tamil Nadu | 0 |
|  | | |  | | | | | |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| F01565 | CORK INTERNATIONAL PTY LIMITED | ACTV | NA | NA | NA | NA | Tamil Nadu | 0 |
| F01566 | ERBIS ENGG COMPANY LIMITED | ACTV | NA | NA | NA | NA | Tamil Nadu | 0 |
| F01589 | RALF SCHNEIDER HOLDING GMBH | NAEF | NA | NA | NA | NA | Tamil Nadu | 0 |
| F01593 | MITRAJAYA TRADING PRIVATE LIMITED | ACTV | NA | NA | NA | NA | Tamil Nadu | 0 |
| F01618 | HEAT AND CONTROL PTY LIMITED | ACTV | NA | NA | NA | 13-07-1999 | Tamil Nadu | 0 |
| F01628 | DIREX SYSTEMS LIMITED | ACTV | NA | NA | NA | NA | Tamil Nadu | 0 |
| F01641 | NMB-MINEBEA THAI LIMITED | NAEF | NA | NA | NA | NA | Tamil Nadu | 0 |
| F01643 | ARROW INTERNATIONAL INC | ACTV | NA | NA | NA | ######## | Tamil Nadu | 0 |
| F01694 | GAMBRO CHINA LTD | ACTV | NA | NA | NA | 14-06-2000 | Tamil Nadu | 0 |
|  |  |  |  |  |  |  |  |  |
| F01703 | OBARA CORPORATION | NAEF | NA | NA | NA | 17-07-2000 | Tamil Nadu | 0 |
| F01752 | CIPTA WAWASON MAJU ENGINEERING SDM BHD | ACTV | NA | NA | NA | 24-01-2001 | Tamil Nadu | 0 |
| F01753 | AUCHAN INTERNATIONAL S.A. | ACTV | NA | NA | NA | NA | Tamil Nadu | 0 |
| F01767 | TOSHIBA PLANT SYSTEMS AND SERVICES CORPORATION | NAEF | NA | NA | NA | 8/3/2001 | Tamil Nadu | 0 |
| F01768 | YAMAZEN CORPORATION | NAEF | NA | NA | NA | NA | Tamil Nadu | 0 |
| F01770 | OWL INTERNATIONAL PTE LTD | ACTV | NA | NA | NA | 22-03-2001 | Tamil Nadu | 0 |
| F01826 | LEXMARK INTERNATIONAL (SINGAPORE) PTE LIMITED | ACTV | NA | NA | NA | 16-08-2001 | Tamil Nadu | 0 |
| F01830 | FLUID ENERGY CONTROLS INC. | ACTV | NA | NA | NA | NA | Tamil Nadu | 0 |
| F01861 | WATCH GUARD TECHNOLOGIES INC | ACTV | NA | NA | NA | 21-11-2001 | Tamil Nadu | 0 |
| F01878 | SINAR JERUIH SDN BHD | ACTV | NA | NA | NA | 24-12-2001 | Tamil Nadu | 0 |
| F01918 | SIPLEC INTERNATIONAL LIMITED | ACTV | NA | NA | NA | 23-09-1995 | Tamil Nadu | 0 |
| F01935 | INTELSAT GLOBAL SERVICES CORPORATION | ACTV | NA | NA | NA | 20-05-2005 | Tamil Nadu | 0 |
| F01940 | PGS GEOPHYSICAL A.S | ACTV | NA | NA | NA | 27-05-2002 | Tamil Nadu | 0 |
| F01987 | SEVERN GLOCON LIMITED | ACTV | NA | NA | NA | 29-08-2002 | Tamil Nadu | 0 |
| F02028 | LAGERWEY WINDTURBINE B V | ACTV | NA | NA | NA | 24-10-2002 | Tamil Nadu | 0 |
| F02061 | SOCAM MANAGEMENT SERVICES SINGAPORE PTELIMITED | NAEF | NA | NA | NA | NA | Tamil Nadu | 0 |
| F02098 | JAN DE NUL NV | ACTV | NA | NA | NA | NA | Tamil Nadu | 0 |
| F02104 | BUCKMAN LABORATORIES (ASIA) PTE. LIMITED | ACTV | NA | NA | NA | 5/2/2003 | Tamil Nadu | 0 |
|  |  |  |  |  |  |  |  |  |
| F02110 | ZWICK ASIA PTE LIMITED | ACTV | NA | NA | NA | 13-02-2002 | Tamil Nadu | 0 |
| F02122 | INVE THAILAND LIMITED | NAEF | NA | NA | NA | NA | Tamil Nadu | 0 |
| F02126 | SUNLEY FASHIONS FAR EAST LIMITED | ACTV | NA | NA | NA | ######## | Tamil Nadu | 0 |
| F02143 | ROTHE ERDE GMBH | NAEF | NA | NA | NA | NA | Tamil Nadu | 0 |
|  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |
| F02157 | RANGASWAMY AND ASSOCIATES INC | ACTV | NA | NA | NA | NA | Tamil Nadu | 0 |
| F02189 | EASTMAN FILMS INC | ACTV | NA | NA | NA | 18-08-2003 | Tamil Nadu | 0 |
| F02222 | XAMBALA INCORPORATED | NAEF | NA | NA | NA | NA | Tamil Nadu | 0 |
| F02235 | DAINTEE LIMITED | ACTV | NA | NA | NA | ######## | Tamil Nadu | 0 |
| F02253 | COLUMBIA SPORTSWEAR COMPANY | ACTV | NA | NA | NA | NA | Tamil Nadu | 0 |
|  |  |  |  |  |  |  |  |  |
| F02261 | KISTLER INSTRUMENTS PTE LIMITED | NAEF | NA | NA | NA | NA | Tamil Nadu | 0 |
| F02262 | AJINOMOTO CO INC | NAEF | NA | NA | NA | 21-01-2004 | Tamil Nadu | 0 |
| F02297 | DANKOTUWA PROCELAIN LIMITED | ACTV | NA | NA | NA | 15-04-2004 | Tamil Nadu | 0 |
| F02337 | PUNCAK NAGA HOLDINGS BERHAD | ACTV | NA | NA | NA | 26-07-2004 | Tamil Nadu | 0 |
| F02339 | SIGMA CORPORATION | NAEF | NA | NA | NA | NA | Tamil Nadu | 0 |
| F02372 | CARGO COMMUNITY NETWORK PTE LTD | ACTV | NA | NA | NA | NA | Tamil Nadu | 0 |
| F02378 | HETTIGODA DISTRIBUTORS PRIVATE LIMITED | ACTV | NA | NA | NA | 17-09-2004 | Tamil Nadu | 0 |
| F02394 | PROPLUS SYSTEMS INC | ACTV | NA | NA | NA | NA | Tamil Nadu | 0 |
| F02418 | DEUTSCHE WOOLWORTH SOURCING HK LIMITED | ACTV | NA | NA | NA | NA | Tamil Nadu | 0 |

**2.Data Preprocessing:**

Cleaning and preprocessing data is a crucial step in the data preparation process before you can use it for machine learning or analysis. Below are the steps you can follow to clean and preprocess your data, including handling missing values and converting categorical features into numerical representations.

1. **Import Libraries**

Start by importing the necessary Python libraries for data manipulation and preprocessing, such as Pandas, NumPy, and Scikit-Learn.

python

import pandas as pd

import numpy as np

from sklearn.preprocessing import LabelEncoder, OneHotEncoder

from sklearn.impute import SimpleImputer

**2. Load Your Dataset** Load your dataset into a Pandas DataFrame. Replace 'your\_data.csv' with the actual file path or URL of your dataset.

python

data = pd.read\_csv('your\_data.csv')

**3. Handling Missing Values** Deal with missing values in your dataset. Depending on the nature of the data, you can choose one of the following methods:

* **Imputation with Mean/Median/Mode**: Fill missing values with the mean, median, or mode of the respective column.

python

imputer = SimpleImputer(strategy='mean') # You can also use 'median' or 'most\_frequent'

data['column\_name'] = imputer.fit\_transform(data[['column\_name']])

* **Dropping Rows**: Remove rows with missing values if the number of missing values is small and doesn't significantly affect your dataset.

python

data.dropna(inplace=True)

**4. Handling Categorical Features**

If your dataset contains categorical features, you need to convert them into numerical representations. This can be done in several ways:

* **Label Encoding**: Use label encoding to convert categorical variables into ordinal integers. This is suitable when there is an ordinal relationship between categories.

python

label\_encoder = LabelEncoder()

data['categorical\_column'] = label\_encoder.fit\_transform(data['categorical\_column'])

* **One-Hot Encoding**: Use one-hot encoding to convert categorical variables into binary columns. Each category becomes a new binary column with 0s and 1s.

python

one\_hot\_encoder = OneHotEncoder()

encoded\_categories = one\_hot\_encoder.fit\_transform(data[['categorical\_column']]).toarray()

encoded\_df = pd.DataFrame(encoded\_categories, columns=one\_hot\_encoder.get\_feature\_names(['categorical\_column']))

data = pd.concat([data, encoded\_df], axis=1)

data.drop(['categorical\_column'], axis=1, inplace=True)

**5. Standardization or Normalization (if necessary)**

Depending on your machine learning algorithm, you might want to standardize or normalize your numerical features to have a consistent scale. You can use techniques like Min-Max scaling or StandardScaler from Scikit-Learn.

python

from sklearn.preprocessing import StandardScaler, MinMaxScaler

scaler = StandardScaler() # or MinMaxScaler

data[['numerical\_column1', 'numerical\_column2']] = scaler.fit\_transform(data[['numerical\_column1', 'numerical\_column2']])

**6. Save Processed Data (Optional)**

If you want to save your cleaned and preprocessed data for future use, you can use the to\_csv method in Pandas or other appropriate file formats.

python

data.to\_csv('preprocessed\_data.csv', index=False)

By following these steps, you can clean and preprocess your data, handle missing values, and convert categorical features into numerical representations suitable for machine learning or analysis. Make sure to customize these steps according to your specific dataset and requirements.

**3.Exploratory Data Analysis:**

Exploratory Data Analysis (EDA) is a crucial step in understanding your data and extracting valuable insights from it. In this example, we'll assume you have a dataset containing information about registered companies. Here's how you can perform EDA to understand the distribution, relationships, and unique characteristics of these companies:

**1. Import Libraries** Start by importing the necessary Python libraries for data analysis and visualization.

python

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

**2. Load Your Dataset** Load your dataset into a Pandas DataFrame if you haven't already (you can reuse the data DataFrame from the previous example).

python

data = pd.read\_csv('your\_data.csv')

**3. Basic Data Exploration**

* **Preview Data**: Use data.head() to display the first few rows of your dataset to get an initial sense of the data's structure.

python

print(data.head())

* **Summary Statistics**: Get summary statistics for numerical columns to understand central tendencies and spreads.

python

print(data.describe())

**4. Data Visualization**

* **Histograms**: Create histograms to visualize the distribution of numerical variables.

python

data['numerical\_column'].plot(kind='hist', bins=20, edgecolor='k')

plt.xlabel('Numerical Column')

plt.ylabel('Frequency')

plt.title('Histogram of Numerical Column')

plt.show()

* **Box Plots**: Use box plots to identify outliers and understand the distribution of numerical variables.

python

sns.boxplot(x='categorical\_column', y='numerical\_column', data=data)

plt.xlabel('Categorical Column')

plt.ylabel('Numerical Column')

plt.title('Box Plot of Numerical Column by Category')

plt.xticks(rotation=90)

plt.show()

* **Count Plots**: Create count plots to visualize the distribution of categorical variables.

python

sns.countplot(x='categorical\_column', data=data)

plt.xlabel('Categorical Column')

plt.ylabel('Count')

plt.title('Count Plot of Categorical Column')

plt.xticks(rotation=90)

plt.show()

**5. Relationships and Correlations**

* **Correlation Matrix**: Compute and visualize the correlation between numerical variables.

python

correlation\_matrix = data.corr()

sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', linewidths=0.5)

plt.title('Correlation Matrix')

plt.show()

* **Pairplots**: Create pairplots to visualize pairwise relationships between numerical variables.

python

sns.pairplot(data, hue='categorical\_column')

plt.suptitle('Pairplot of Numerical Variables')

plt.show()

**6. Unique Characteristics**

* **Unique Values**: Explore the unique values in categorical columns to identify unique characteristics.

python

unique\_values = data['categorical\_column'].unique()

print("Unique Values in Categorical Column:", unique\_values)

* **Value Counts**: Get the count of each unique value in a categorical column.

python

value\_counts = data['categorical\_column'].value\_counts()

print("Value Counts:\n", value\_counts)

These are some common EDA techniques to get a better understanding of your data. You can customize and expand your analysis based on the specific questions you want to answer and the characteristics of your

**4.Feature engineering:**

Feature engineering involves creating new features or transforming existing ones to improve the performance of predictive models. The goal is to provide the model with more relevant and informative input data. Here are some techniques and examples for feature engineering:

**1. Encoding Categorical Variables:**

* We've discussed this in the data preprocessing section. You can use techniques like one-hot encoding or label encoding to convert categorical variables into numerical representations.

**2. Date and Time Features:**

* Extract meaningful information from date and time variables such as year, month, day, day of the week, or time of day. These can be useful in time-series analysis or when time-related patterns matter.

python

data['year'] = data['date'].dt.year

data['month'] = data['date'].dt.month

data['day\_of\_week'] = data['date'].dt.dayofweek

**3. Aggregation and Summary Statistics:**

* Create new features by aggregating or summarizing existing ones. For example, calculate the mean, sum, or standard deviation of numerical variables for each category in a categorical column.

python

# Calculate the mean of a numerical column for each category in a categorical column

mean\_by\_category = data.groupby('categorical\_column')['numerical\_column'].mean()

data['mean\_numerical\_by\_category'] = data['categorical\_column'].map(mean\_by\_category)

**4. Interaction Features:**

* Create new features by combining existing ones to capture interactions or relationships between them. This can be useful in cases where the interaction has predictive power.

python

data['interaction\_feature'] = data['feature1'] \* data['feature2']

**5. Polynomial Features:**

* Create polynomial features to capture non-linear relationships in the data. This is particularly useful in polynomial regression or when you suspect that higher-order terms are significant.

python

from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2)

X\_poly = poly

**5.Predictive Modelling:**

To develop predictive models for future company registrations, you can follow these steps:

**\*\*1. Data Preparation:\*\***

**- Ensure your dataset is** cleaned, preprocessed, and contains the relevant features as discussed earlier.

- Split your data into training and testing sets to evaluate the model's performance.

```python

from sklearn.model\_selection import train\_test\_split

X = data.drop('target\_variable', axis=1)

y = data['target\_variable']

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

```

2. **Model Selection:\*\***

- Choose appropriate machine learning algorithms based on the nature of your problem. Common choices for predictive modeling include:

- \*\*Linear Regression\*\*: For regression tasks when the target variable is continuous.

- \*\*Logistic Regression\*\*: For binary classification tasks.

- \*\*Random Forest\*\*, \*\*Gradient Boosting\*\*, \*\*XGBoost\*\*: For both regression and classification tasks, and they often perform well.

- \*\*Neural Networks\*\*: For complex problems with large datasets.

- \*\*Support Vector Machines (SVM)\*\*: For classification and regression tasks, especially when dealing with high-dimensional data.

\*\*3. Model Training:\*\*

- Train your chosen machine learning models using the training data.

```python

from sklearn.ensemble import RandomForestClassifier # Replace with the appropriate model

model = RandomForestClassifier() # Initialize the model

model.fit(X\_train, y\_train) # Train the model

```

\*\*4. Model Evaluation:\*\*

- Assess the model's performance using appropriate evaluation metrics. For classification, common metrics include accuracy, precision, recall, F1-score, and ROC-AUC. For regression, you can use metrics like mean squared error (MSE), R-squared, and mean absolute error (MAE).

```python

from sklearn.metrics import accuracy\_score, classification\_report, mean\_squared\_error

# For classification

y\_pred = model.predict(X\_test)

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

report = classification\_report(y\_test, y\_pred)

# For regression

y\_pred = model.predict(X\_test)

mse = mean\_squared\_error(y\_test, y\_pred)

```

\*\*5. Hyperparameter Tuning:\*\*

- Optimize your model's hyperparameters to improve its performance. You can use techniques like Grid Search or Random Search.

```python

from sklearn.model\_selection import GridSearchCV

param\_grid = {'n\_estimators': [100, 200, 300], 'max\_depth': [None, 10, 20]}

grid\_search = GridSearchCV(RandomForestClassifier(), param\_grid, cv=5)

grid\_search.fit(X\_train, y\_train)

best\_params = grid\_search.best\_params\_

**6.Model evaluation:**

Model evaluation is a crucial step in assessing the performance of your predictive models. The choice of evaluation metrics depends on the nature of the problem you are trying to solve (classification, regression, etc.). Below, I'll provide examples of how to evaluate predictive models using common metrics for classification and regression tasks:

**Classification Metrics:**

1. **Accuracy:** It measures the proportion of correctly predicted instances out of the total instances.

python

from sklearn.metrics import accuracy\_score

y\_true = [0, 1, 1, 0, 1]

y\_pred = [0, 1, 0, 0, 1]

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

print("Accuracy:", accuracy)

1. **Precision:** It measures the proportion of true positive predictions among all positive predictions.

python

from sklearn.metrics import precision\_score

precision = precision\_score(y\_true, y\_pred)

print("Precision:", precision)

1. **Recall (Sensitivity or True Positive Rate):** It measures the proportion of true positives correctly predicted among all actual positives.

python

from sklearn.metrics import recall\_score

recall = recall\_score(y\_true, y\_pred)

print("Recall:", recall)

1. **F1-Score:** It is the harmonic mean of precision and recall and is useful when you want to balance precision and recall.

python

from sklearn.metrics import f1\_score

f1 = f1\_score(y\_true, y\_pred)

print("F1-Score:", f1)

1. **Confusion Matrix:** It provides a detailed breakdown of the model's predictions, including true positives, true negatives, false positives, and false negatives.

python

from sklearn.metrics import confusion\_matrix

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

print("Confusion Matrix:\n", conf\_matrix)

1. **Receiver Operating Characteristic (ROC) Curve and Area Under the Curve (AUC):** Useful for binary classification problems with a probability score.

python

from sklearn.metrics import roc\_curve, roc\_auc\_score

y\_probs = model.predict\_proba(X\_test)[:, 1]

fpr, tpr, thresholds = roc\_curve(y\_true, y\_probs)

roc\_auc = roc\_auc\_score(y\_true, y\_probs)

# Plot ROC Curve

plt.figure(figsize=(8, 6))

plt.plot(fpr, tpr, label='ROC curve (area = {:.2f})'.format(roc\_auc))

plt.plot([0, 1], [0, 1], 'k--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC)')

plt.legend(loc='lower right')

plt.show()

**Regression Metrics:**

1. **Mean Absolute Error (MAE):** It measures the average absolute difference between predicted and actual values.

python

from sklearn.metrics import mean\_absolute\_error

y\_true = [3.0, 4.5, 2.0, 5.1, 6.3]

y\_pred = [2.8, 4.2, 2.2, 5.0, 6.0]

mae = mean\_absolute\_error(y\_true, y\_pred)

print("MAE:", mae)

1. **Mean Squared Error (MSE):** It measures the average of the squared differences between predicted and actual values.

python

from sklearn.metrics import mean\_squared\_error

mse = mean\_squared\_error(y\_true, y\_pred)

print("MSE:", mse)

1. **Root Mean Squared Error (RMSE):** It is the square root of MSE and provides the error in the same units as the target variable.

python

import numpy as np

rmse = np.sqrt(mse)

print("RMSE:", rmse)

1. **R-squared (R2):** It measures the proportion of the variance in the dependent variable that is predictable from the independent variables.

python

from sklearn.metrics import r2\_score

r2 = r2\_score(y\_true, y\_pred)

print("R-squared:", r2)

When evaluating predictive models, choose the evaluation metrics that are most relevant to your specific problem and consider the trade-offs between them. It's often a good practice to use a combination of metrics to get a comprehensive view of the model's performance.