

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

Team Member

510521104009: EZHILARASAN R

PHASE-1:DOCUMENTSUBMISSION



OBJECTIVE:

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>

1.DataSource;

F00643	HOCHTIEFFAG,	NAEF	NA	NA	NA	#####	Tamil Nadu	0
F00721	SUMITOMO CORPORATION(SUMITOM OSHOJKAISHA LIMITED)	ACTV	NA	NA	NA	NA	Tamil Nadu	0
F00892	SRILANKANAIRLINESLIMITED	ACTV	NA	NA	NA	1/3/1982	Tamil Nadu	0
F01208	CALTEXINDIALIMITED	NAEF	NA	NA	NA	NA	Tamil Nadu	0
F01218	GEHEALTHCAREBIO-SCIENCES LIMITED	ACTV	NA	NA	NA	NA	Tamil Nadu	0
F01265	CAIRNENERGYINDIAPTY. LIMITED	NAEF	NA	NA	NA	NA	Tamil Nadu	0
F01269	TORIELLIS.R.L	ACTV	NA	NA	NA	5/9/1995	Tamil Nadu	0
F01311	HARDYEXPLORATION& PRODUCTION(INDIA)INC..	ACTV	NA	NA	NA	NA	Tamil Nadu	0
F01314	HOCHTIOFAKTIENGESELLSHARFF VORMGFBRHELFMANN	ACTV	NA	NA	NA	#####	Tamil Nadu	0
F01412	EPSONSINGAPOREPVT LTD	ACTV	NA	NA	NA	25-04-1997	Tamil Nadu	0
F01426	CARGOLUXAIRLINES INTERNATIONALS A	ACTV	NA	NA	NA	#####	Tamil Nadu	0
F01468	CHOHEUNGELECTRIC INDUSTRIALCOMPANYLIMITED	NAEF	NA	NA	NA	NA	Tamil Nadu	0
F01543	NYCOMED ASIAPACIFICPTE LIMITED	ACTV	NA	NA	NA	27-10-1998	Tamil Nadu	0
F01544	CHERRINGTONASIALTD	ACTV	NA	NA	NA	1/5/2000	Tamil Nadu	0
F01563	SHIMADZUASIAPACIFICPTE LIMITED	NAEF	NA	NA	NA	NA	Tamil Nadu	0

F01565	CORKINTERNATIONALPTYLIMITED	ACTV	NA	NA	NA	NA	Tamil Nadu	0
F01566	ERBISENGGCOMPANYLIMITED	ACTV	NA	NA	NA	NA	Tamil Nadu	0
F01589	RALFSCHNEIDERHOLDINGGMBH	NAEF	NA	NA	NA	NA	Tamil Nadu	0
F01593	MITRAJAYATRADINGPRIVATE LIMITED	ACTV	NA	NA	NA	NA	Tamil Nadu	0
F01618	HEATANDCONTROLPTYLIMITED	ACTV	NA	NA	NA	13-07-1999	Tamil Nadu	0
F01628	DIREXSYSTEMSLIMITED	ACTV	NA	NA	NA	NA	Tamil Nadu	0
F01641	NMB-MINEBEATHAILIMITED	NAEF	NA	NA	NA	NA	Tamil Nadu	0

F01643	ARROWINTERNATIONALINC	ACTV	NA	NA	NA	#####	Tamil Nadu	0
F01694	GAMBROCHINA LTD	ACTV	NA	NA	NA	14-06-2000	Tamil Nadu	0
F01703	OBARACORPORATION	NAEF	NA	NA	NA	17-07-2000	Tamil Nadu	0
F01752	CIPTAWAWASONMAJU ENGINEERINGSDBMBHD	ACTV	NA	NA	NA	24-01-2001	Tamil Nadu	0
F01753	AUCHANINTERNATIONALS.A.	ACTV	NA	NA	NA	NA	Tamil Nadu	0
F01767	TOSHIBAPLANTSYSTEMSAND SERVICESCORPORATION	NAEF	NA	NA	NA	8/3/2001	Tamil Nadu	0
F01768	YAMAZENCORPORATION	NAEF	NA	NA	NA	NA	Tamil Nadu	0
F01770	OWLINTERNATIONALPTELTD	ACTV	NA	NA	NA	22-03-2001	Tamil Nadu	0
F01826	LEXMARKINTERNATIONAL (SINGAPORE)PTELIMITED	ACTV	NA	NA	NA	16-08-2001	Tamil Nadu	0
F01830	FLUIDENERGYCONTROLSINC.	ACTV	NA	NA	NA	NA	Tamil Nadu	0
F01861	WATCHGUARDTECHNOLOGIES INC	ACTV	NA	NA	NA	21-11-2001	Tamil Nadu	0
F01878	SINARJERUIHSDNBHD	ACTV	NA	NA	NA	24-12-2001	Tamil Nadu	0
F01918	SIPLECINTERNATIONALLIMITED	ACTV	NA	NA	NA	23-09-1995	Tamil Nadu	0
F01935	INTELSATGLOBALESERVICES CORPORATION	ACTV	NA	NA	NA	20-05-2005	Tamil Nadu	0
F01940	PGSGEOPHYSICALA.S	ACTV	NA	NA	NA	27-05-2002	Tamil Nadu	0
F01987	SEVERNGLONLIMITED	ACTV	NA	NA	NA	29-08-2002	Tamil Nadu	0
F02028	LAGERWEYWINDTURBINEBV	ACTV	NA	NA	NA	24-10-2002	Tamil Nadu	0
F02061	SOCAMMANAGEMENTSERVICES SINGAPOREPTELIMITED	NAEF	NA	NA	NA	NA	Tamil Nadu	0
F02098	JANDENULNV	ACTV	NA	NA	NA	NA	Tamil Nadu	0
F02104	BUCKMANLABORATORIES(ASIA) PTE.LIMITED	ACTV	NA	NA	NA	5/2/2003	Tamil Nadu	0
F02110	ZWICKASIAPTELIMITED	ACTV	NA	NA	NA	13-02-2002	Tamil Nadu	0
F02122	INVETHAILANDLIMITED	NAEF	NA	NA	NA	NA	Tamil Nadu	0

F02126	SUNLEYFASHIONSFAREAST LIMITED	ACTV	NA	NA	NA	#####	Tamil Nadu	0
F02143	ROTHERDEGMBH	NAEF	NA	NA	NA	NA	Tamil Nadu	0
F02157	RANGASWAMYANDASSOCIATES INC	ACTV	NA	NA	NA	NA	Tamil Nadu	0
F02189	EASTMANFILMSINC	ACTV	NA	NA	NA	18-08-2003	Tamil Nadu	0
F02222	XAMBALAINCORPORATED	NAEF	NA	NA	NA	NA	Tamil Nadu	0
F02235	DAINTEELIMITED	ACTV	NA	NA	NA	#####	Tamil Nadu	0
F02253	COLUMBIA SPORTSWEARCOMPANY	ACTV	NA	NA	NA	NA	Tamil Nadu	0
F02261	KISTLER INSTRUMENTS PTELIMITED	NAEF	NA	NA	NA	NA	Tamil Nadu	0
F02262	AJINOMOTOCOINC	NAEF	NA	NA	NA	21-01-2004	Tamil Nadu	0
F02297	DANKOTUWAPROCELAINLIMITED	ACTV	NA	NA	NA	15-04-2004	Tamil Nadu	0
F02337	PUNCAK NAGAHOLDINGSBERHAD	ACTV	NA	NA	NA	26-07-2004	Tamil Nadu	0
F02339	SIGMACORPORATION	NAEF	NA	NA	NA	NA	Tamil Nadu	0
F02372	CARGOCOMMUNITYNETWORK PTE LTD	ACTV	NA	NA	NA	NA	Tamil Nadu	0
F02378	HETTIGODADISTRIBUTORS PRIVATELIMITED	ACTV	NA	NA	NA	17-09-2004	Tamil Nadu	0
F02394	PROPLUSSYSTEMSINC	ACTV	NA	NA	NA	NA	Tamil Nadu	0
F02418	DEUTSCHEWOOLWORTH SOURCINGHKLIMITED	ACTV	NA	NA	NA	NA	Tamil Nadu	0

2.DataPreprocessing:

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.ExploratoryDataAnalysis:

ExploratoryDataAnalysis(EDA)isacrucialstepinunderstandingyourdata and extracting valuable insights from it. In this example, we'llassume you have a dataset containing information about registeredcompanies. Here's how you can perform EDA to understand thedistribution,relationships,anduniquecharacteristicsofthesecompanies:

1. ImportLibraries

StartbyimportingthenecessaryPythonlibrariesfordataanalysis and visualization.

```
python
importpandasaspd
importnumpyasnp
importmatplotlib.pyplotasplt
importseabornas sns
```

2. LoadYourDatasetLoadyourdatasetintoaPandasDataFrameifyou haven't already (you can reuse the data DataFrame from thepreviousexample).

```
python
data=pd.read_csv('your_data.csv')
```

3. BasicDataExploration

- **PreviewData:**Usedata.head()todisplaythefirstfew rowsofyourdatasettogetaninitialsenseof thedata'sstructure.

```
pythonprint(data.head())
```

- **SummaryStatistics:**Getsummarystatisticsfornumericalcolumnstounderstandcentraltendenciesandspreads.

```
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('NumericalColumn')
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('CategoricalColumn')
plt.ylabel('NumericalColumn')
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('CategoricalColumn')
plt.ylabel('Count')
plt.title('Count Plot of Categorical Column')
plt.xticks(rotation=90)
plt.show()
```

5. Relationships and Correlations

- **CorrelationMatrix:** 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('CorrelationMatrix')
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

- **UniqueValues:** Explore the unique values in categorical columns to identify unique characteristics.

```
python
unique_values =
data['categorical_column'].unique()print("UniqueValuesinCategoricalColumn:",unique_values)
```

- **ValueCounts:**
Get the count of each unique value in a categorical column.

```
python
value_counts=data['categorical_column'].value_counts()print("ValueCounts:\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_pr
ed)
```

```
#For regression
y_pred=model.predict(X_test)
mse=mean_squared_error(y_test,y_pred)
```
```

****5.HyperparameterTuning:****

- Optimize your model's hyperparameters to improve its performance. You can use techniques like GridSearch or RandomSearch.

```
```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)
```

2. **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)
```

3. **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)
```



4. **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)
```

5. **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)
```

6. **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)
```

2. **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)
```

3. **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)
```

4. **R-squared (R<sup>2</sup>):** 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.