**AI-DRIVEN EXPLORATION AND PREDICTION OF COMPANY REGISTRATION TRENDS WITH REGISTRAR OF COMPANIES (RoC)**

**INSERTING LIBRARIES:**import numpy as np

import pandas as pd

import seaborn as sns

import os

import matplotlib as plt

import matplotlib.pyplot as plt

**numpy (np):**

For handling large, multi-dimensional arrays and matrices, and performing mathematical functions on them.

**pandas (pd):**

Provides high-level data structures and tools for data analysis, including DataFrames for in-memory 2D tables.

**seaborn (sns):**

A visualization library for statistical plotting, building on matplotlib and integrating with pandas.

os:

Offers OS-dependent functionality, such as interfacing with the file system.

**matplotlib (plt):**

Note: The initial import is redundant. The library is used for creating a wide range of static and interactive visualizations.

**matplotlib.pyplot (plt):**

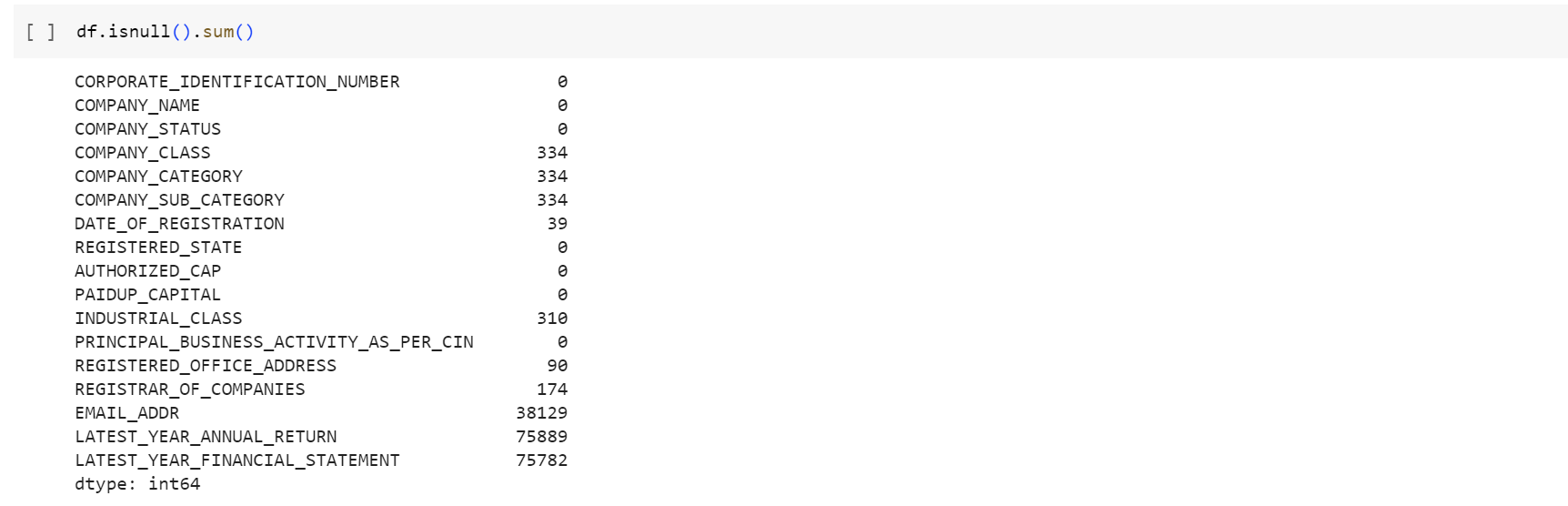
A submodule in matplotlib that offers a state-based interface for creating visualizations.

These libraries collectively support various stages of data manipulation, analysis, and visualization in Python.

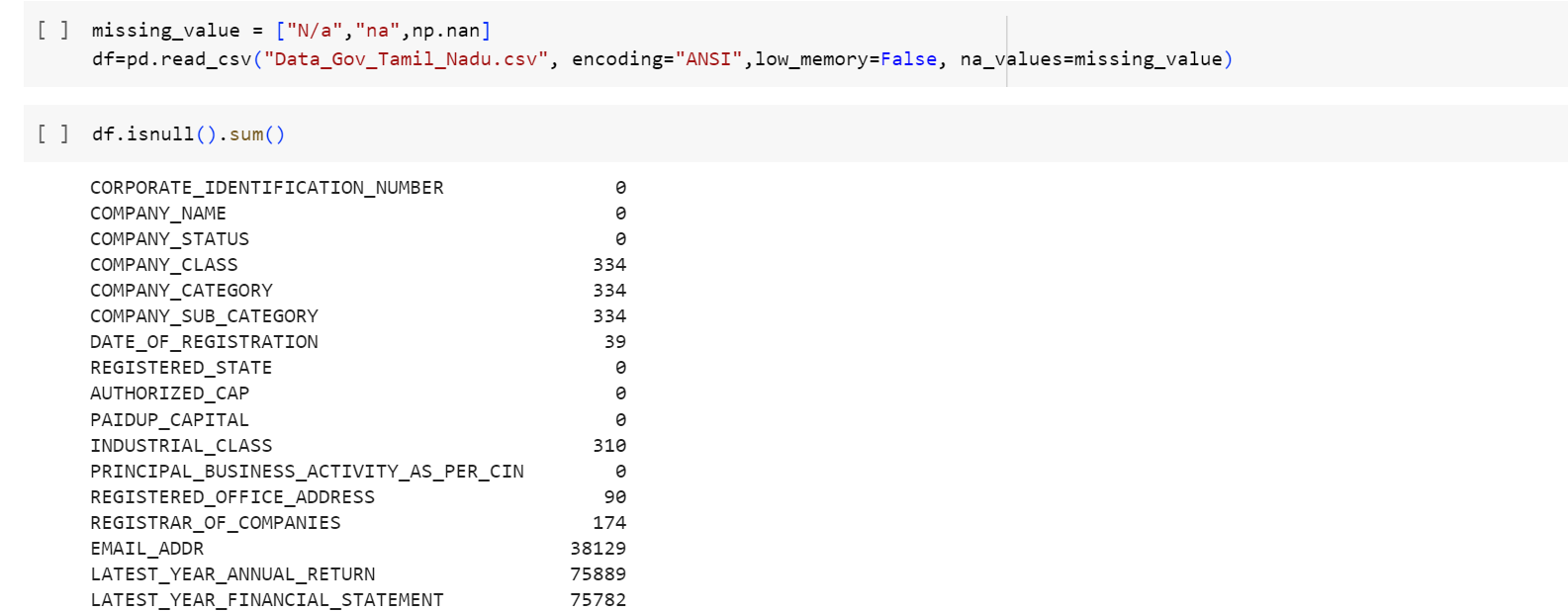
**LOADING CSV DATASET**

****

**DISPLAYING NULL VALUES**

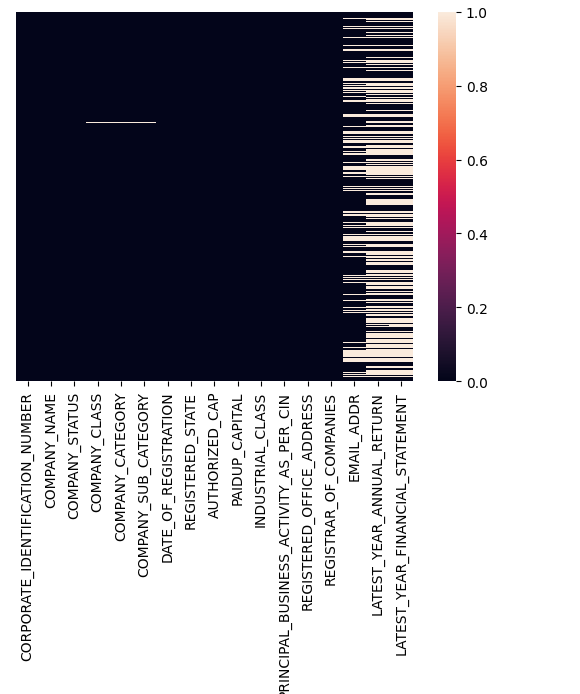


**HANDLING NULL VALUES**



na\_values=missing\_value. This means that while reading the CSV, if any of the values in the list missing\_value are encountered, they will be treated as NaN (Not a Number) values, which is a standard way of representing missing data in pandas.

**VISUALISATION OF NULL VALUES USING A HEAT MAP**





This line drops rows from the DataFrame df where all of its elements are NaN (missing values). The resulting DataFrame after dropping such rows is stored in df\_dropped.

The dropna() method is used to drop missing values.

**DROPPING DUPLICATE VALUES**

df\_dupfree =df.drop\_duplicates(["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"])

This line removes duplicate rows from the df DataFrame based on a specified set of columns. The drop\_duplicates() method is used to drop duplicate rows.

The columns passed inside the method specify the criteria for determining duplicates. In this case, a row is considered a duplicate if all the mentioned columns have the same values as another row.

# HANDLING MISSING DATA

# For Categorical Columns:

## Handling missing values in Date\_Of\_Registration

# 

## Handling missing values in Industrial\_Class



# .fillna() is a pandas method to fill NA/NaN values using the specified method. In this case, the method chosen is to fill missing values with the mode of the column.

# Here, mode() calculates the mode of the 'COMPANY\_CLASS' column. The mode is the value that appears most frequently in a column. Since the mode() method returns a Series (because a column can technically have more than one mode), [0] is used to select the first mode.

# inplace=True ensures that the changes are made directly to the df\_dupfree DataFrame without the need to assign the result back to df\_dupfree.

# Fill missing values in the REGISTERED\_OFFICE\_ADDRESS and REGISTRAR\_OF\_COMPANIES column with "UNKNOWN"

# 

This line replaces missing values in the columns REGISTERED\_OFFICE\_ADDRESS, REGISTRAR\_OF\_COMPANIES, of the df\_dupfree DataFrame with the string "UNKNOWN".

## Count of Null values after data handling

## 

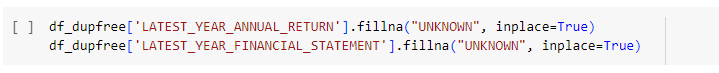
## 

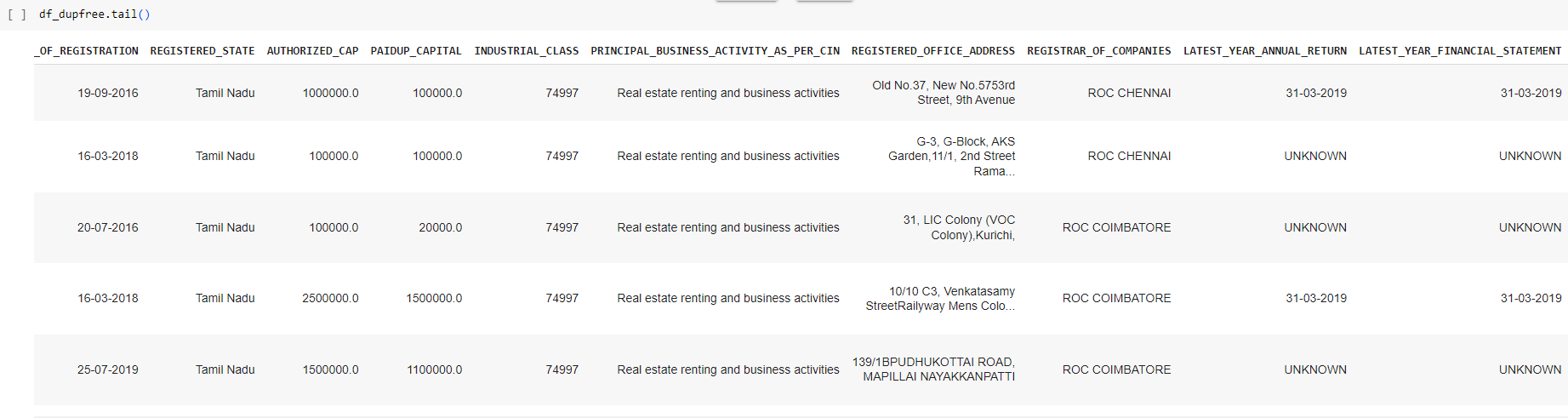
## We should handle the columns EMAIL\_ADDR, LATEST\_YEAR\_ANNUAL\_RETURN and LATEST\_YEAR\_FINANCIAL\_STATEMENT. Every time, after handling a column, check changes to the dataset which will help in deciding what to do next.

## DROPPING IRRELEVANT DATA

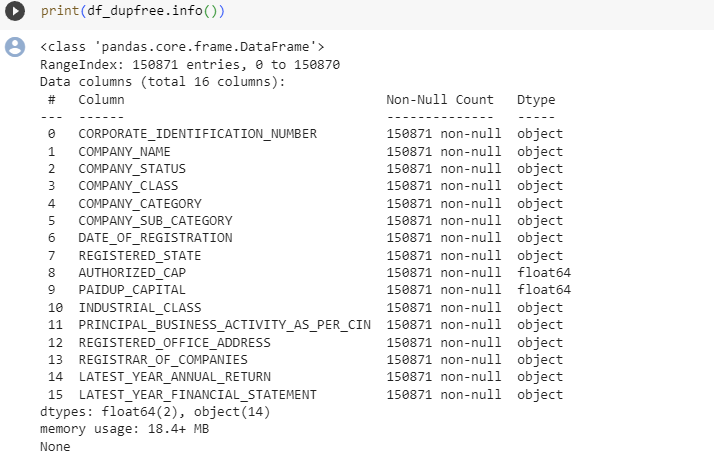
This line  drops the EMAIL\_ADDR column, as it is not relevant to the analysis.

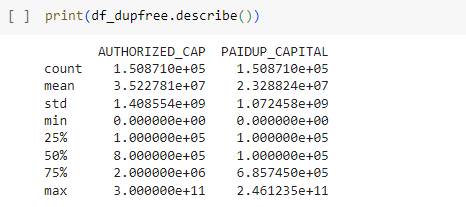
## Fill missing values in the LATEST\_YEAR\_ANNUAL\_RETURN and LATEST\_YEAR\_FINANCIAL\_STATEMENT column with "UNKNOWN"

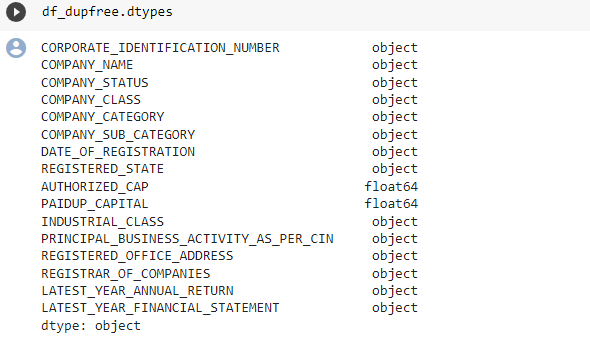




**GET AN OVERVIEW OF DATA**

****

**SUMMARY STATISTICS  
**

****

**Correlation matrix**

****

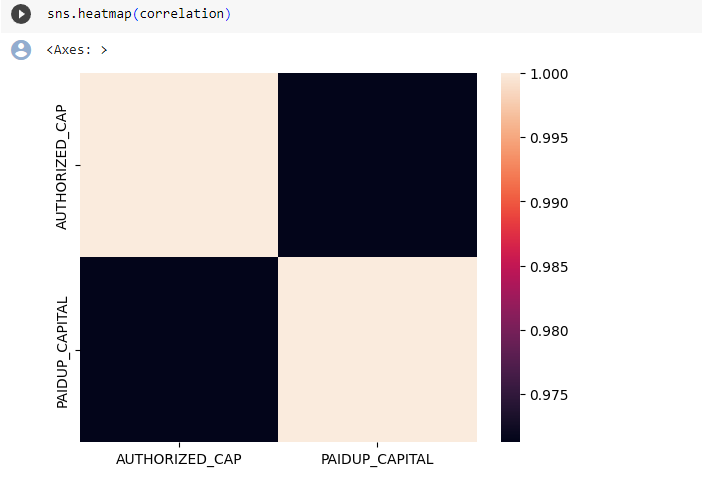
This line calculates the Pearson correlation coefficient between the columns "AUTHORIZED\_CAP" and "PAIDUP\_CAPITAL". The result is a matrix (in the form of a DataFrame) that contains correlation coefficients between the provided columns.

The .corr() method calculates the Pearson correlation, which provides a value between -1 and 1 indicating the extent to which two variables change in relation to each other.

* A value closer to 1 implies a strong positive correlation: as one variable increases, the other also tends to increase.
* A value closer to -1 implies a strong negative correlation: as one variable increases, the other tends to decrease.
* A value closer to 0 implies little to no linear relationship between the variables.

The correlation between these two columns might give insights into how closely companies tend to operate to their authorized limits.

These fields have been chosen specifically because understanding the relationship between authorized and paid-up capital is crucial for financial analysis or understanding the financial health and strategies of companies.

****

This line uses the seaborn library to visualize the correlation matrix as a heatmap. In a heatmap, data values are represented as colors. Darker or lighter colors in the heatmap will represent the strength and direction (positive or negative) of the correlation.

**EXPLORATORY DATA ANALYSIS (EDA)**

## Distribution of company registrations by industry, location, and size

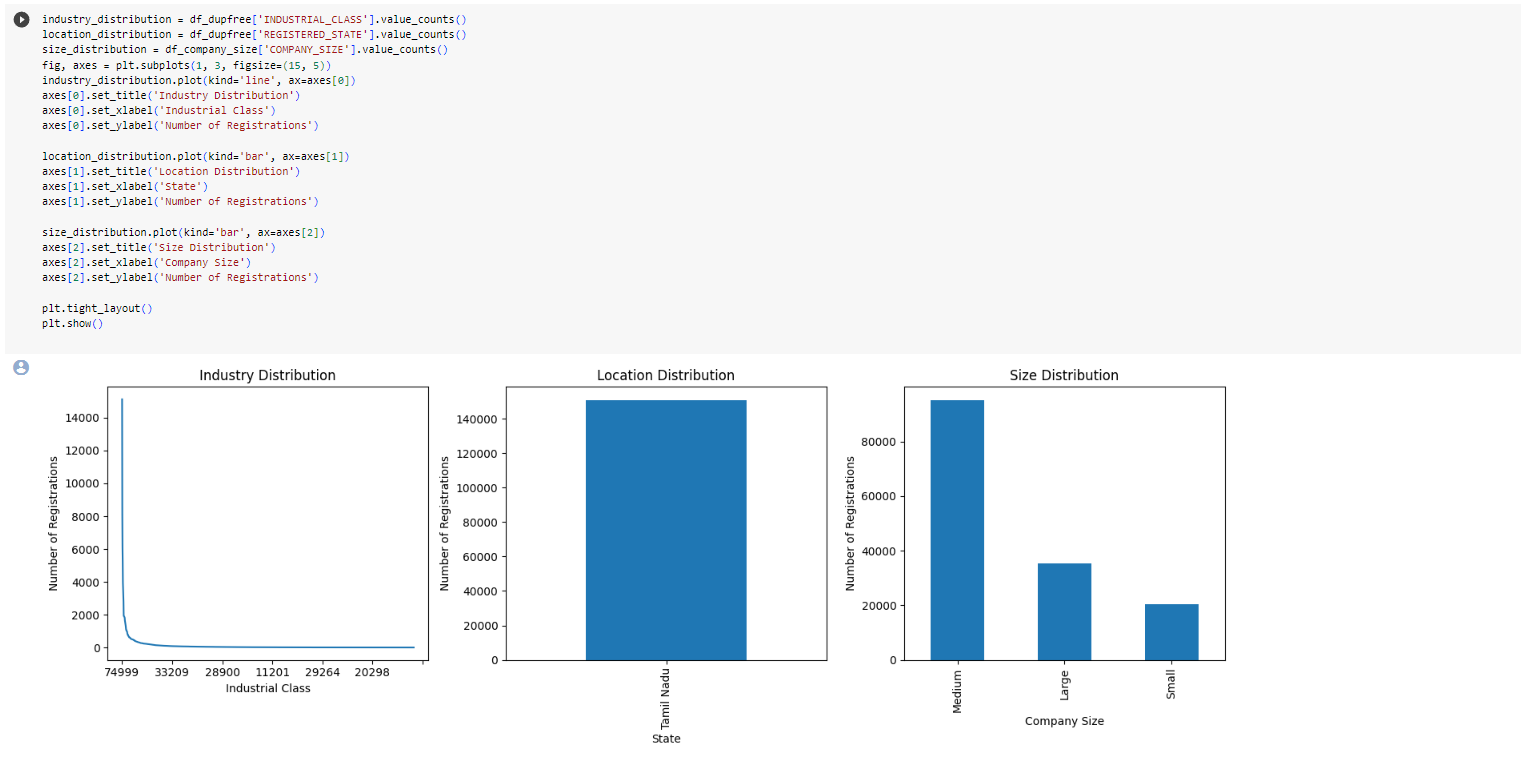


This segment of the code creates a new DataFrame df\_company\_size by extracting the first 150,870 entries for three columns: COMPANY\_NAME, AUTHORIZED\_CAP, and PAIDUP\_CAPITAL from the df\_dupfree DataFrame.

Companies are categorized into 'Small', 'Medium', and 'Large' based on their authorized capital and paid-up capital. Criteria for categorizing are defined in the size\_categories dictionary.

conditions contain boolean expressions for categorizing companies as 'Small' or 'Medium'.

Using np.select(), companies satisfying the conditions are labeled 'Small' or 'Medium'. Any company that doesn't match these conditions defaults to 'Large'.



The distribution (counts) for industry classes, registered states (locations), and company sizes are calculated.

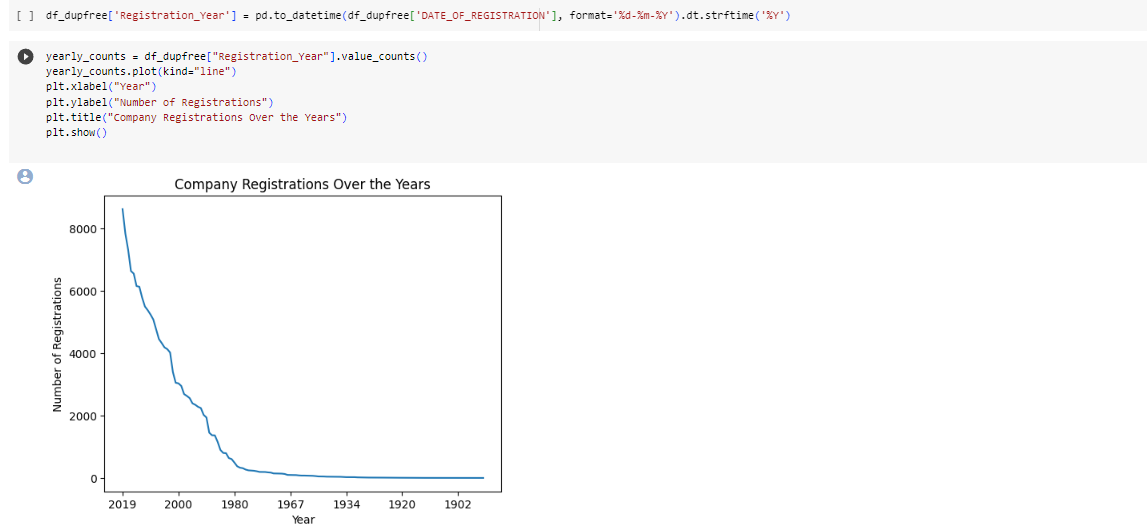
The visualization of the distributions is done using matplotlib.

plt.subplots(1, 3, figsize=(15, 5)): Initializes a single row of three subplots (charts).

plt.tight\_layout() adjusts the spacing between the plots for better aesthetics.

plt.show() displays the visualizations.

## Company Registrations Trends Over Time



pd.to\_datetime(...): The 'DATE\_OF\_REGISTRATION' column, which might be in a string format, is converted to a datetime object.

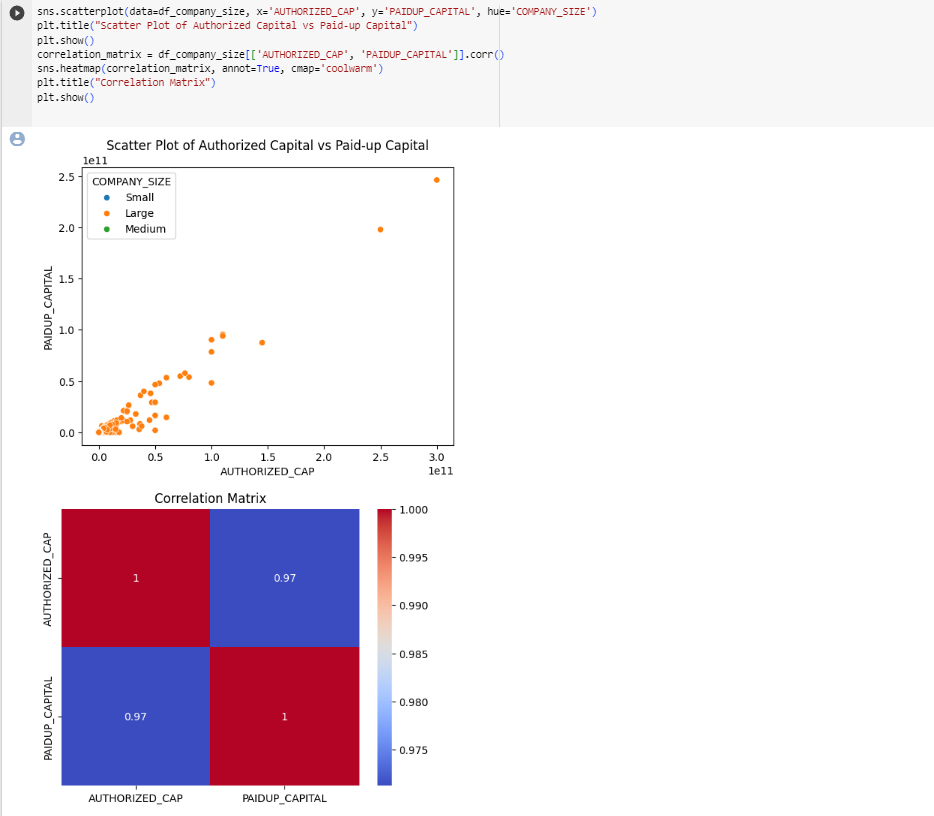
.dt.strftime('%Y'): Extracts only the year from the datetime object.

value\_counts(): Calculates the number of registrations for each year.

The plot displays the yearly registration trend using a line chart.

The x-axis represents the year, while the y-axis shows the number of registrations for that year.

Relationships between different variables, such as company size, authorized capital, and paid-up

****

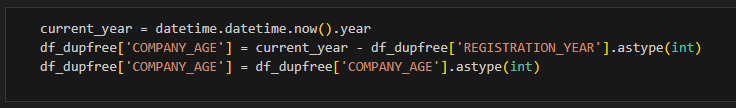
This scatterplot illustrates the relationship between authorized capital and paid-up capital, while the colors (determined by the hue parameter) represent the size of the company.

Correlation matrix is also obtained which helps in investigating relationships between company financial metrics (authorized vs. paid-up capital) and how company size might affect this relationship.

The dataset used is df\_company\_size DataFrame, which is derived from df\_dupfree but only contains the first 150,870 rows and specifically the columns 'COMPANY\_NAME', 'AUTHORIZED\_CAP', and 'PAIDUP\_CAPITAL'. The reason for using this subset may be to focus on a specific segment of the data for more detailed analysis or due to computational efficiency.

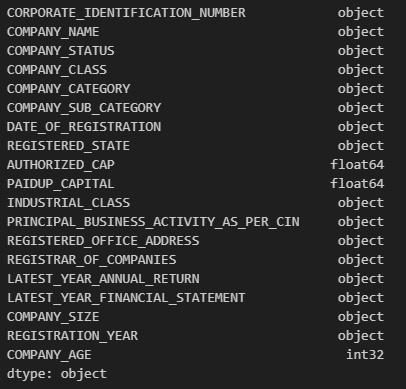
**FEATURE ENGINEERING:**

Age of Companies:

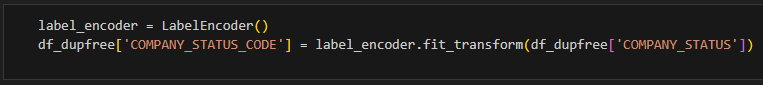


After running this code, the DataFrame df\_dupfree will have a new column called 'COMPANY\_AGE', and each row in this column will represent the age of a company calculated as the difference between the current year and the registration year. This age is expressed in years, as an integer.

This code is useful for creating a feature, 'COMPANY\_AGE,' that can be used in data analysis, machine learning, or any other analysis related to the age of companies based on their registration year.

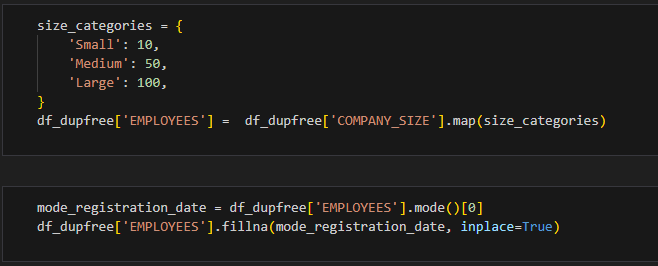


Label Encoding for “COMPANY\_STATUS”:



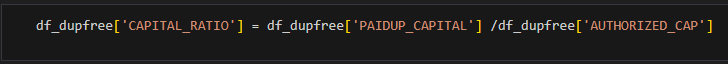
This takes a categorical column, 'COMPANY\_STATUS,' in your DataFrame and converts it into a numerical representation. This is done by assigning a unique integer to each distinct category in the 'COMPANY\_STATUS' column. The LabelEncoder ensures that each unique category corresponds to a specific integer, making it easier to work with in machine learning models or data analysis.

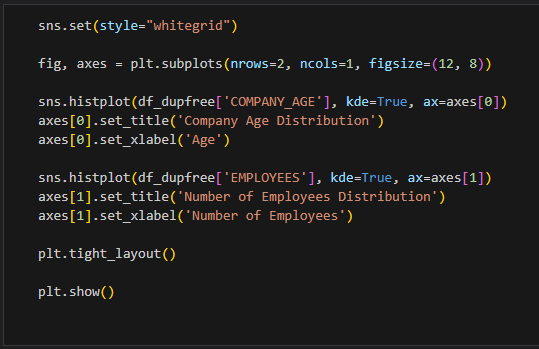
Number of Employees:



It involves mapping the values from 'COMPANY\_SIZE' to new numerical values according to a predefined dictionary, 'size\_categories.'

Calculate the ratio of ‘PAIDUP\_CAPITAL’ to ‘AUTHORIZED\_CAP’:

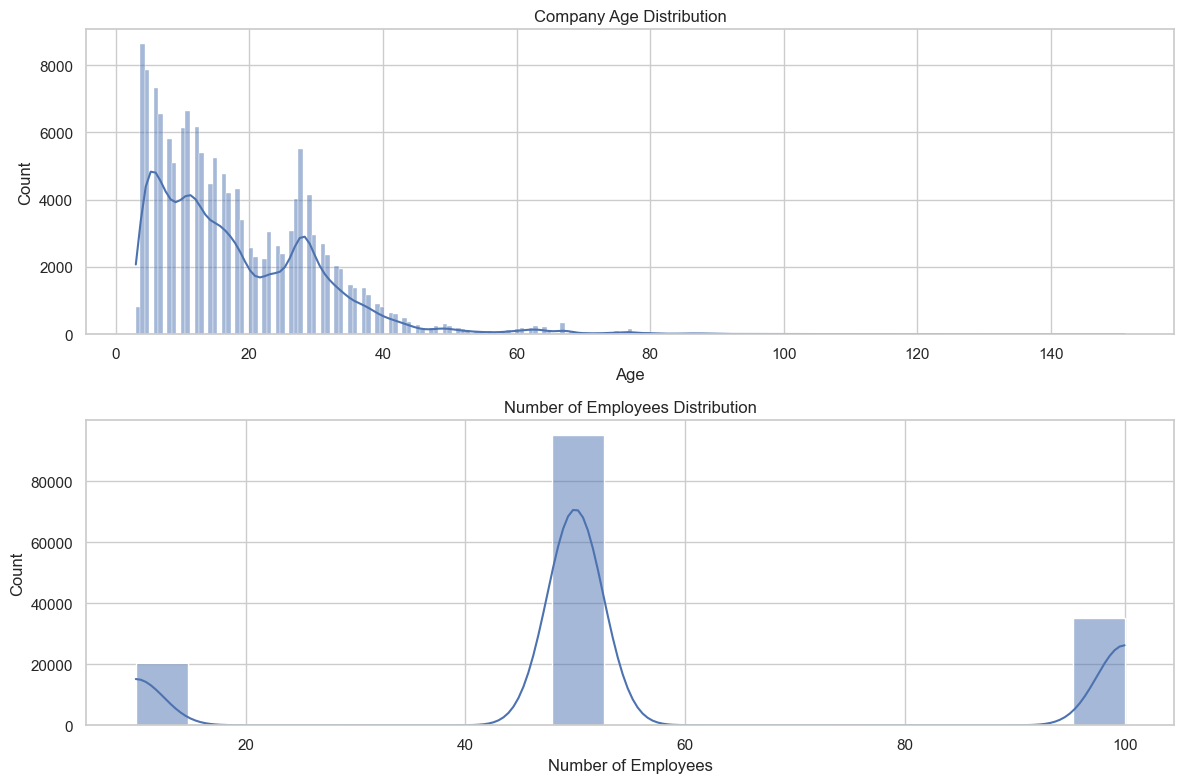




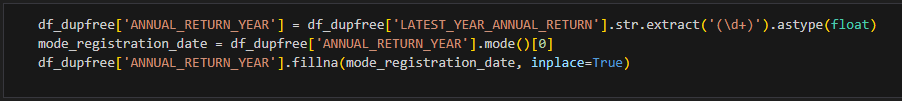
The code computes the ratio of paid-up capital to authorized capital for each company in the DataFrame 'df\_dupfree' by performing element-wise division. Each row in the 'CAPITAL\_RATIO' column will contain the result of dividing the 'PAIDUP\_CAPITAL' value for that company by its corresponding 'AUTHORIZED\_CAP' value.

Reasons for calculating the ratio:

* Financial Analysis
* Investment Decisions
* Regulatory Compliance
* Strategic Decision Making



Extract ‘LATEST\_YEAR\_ANNUAL\_RETURN’ year:



This extracts the year from the 'LATEST\_YEAR\_ANNUAL\_RETURN' column, stores it in a new column 'ANNUAL\_RETURN\_YEAR,' and fills any missing year values with the mode of the available year values. This is a common data preprocessing step when working with date or time-related data to ensure that the data is in a consistent format and that missing values are appropriately handled.

Random Forest Algorithm:

import pandas as pd

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

from sklearn.preprocessing import LabelEncoder

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, classification\_report,confusion\_matrix

import matplotlib.pyplot as plt

import seaborn as sns

df\_dupfree = pd.read\_csv("Data\_Gov\_Tamil\_Nadu.csv", encoding="ANSI",low\_memory=False)

df\_dupfree = df\_dupfree[['COMPANY\_STATUS', 'COMPANY\_CLASS', 'COMPANY\_CATEGORY',

'AUTHORIZED\_CAP',

'PAIDUP\_CAPITAL',

'PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN']]

df\_dupfree.dropna(inplace=True)

df\_dupfree = df\_dupfree.dropna(how="all")

mapping = {

    'Public': 1,

    'Private': 2,

    'Private(One Person Company)': 3,

}

df\_dupfree['COMPANY\_CLASS'] = df\_dupfree['COMPANY\_CLASS'].map(mapping)

mappingC = {

    'Company limited by Shares': 1,

    'Company Limited by Guarantee': 2,

    'Unlimited Company': 3,

}

df\_dupfree['COMPANY\_CATEGORY'] = df\_dupfree['COMPANY\_CATEGORY'].map(mappingC)

mappingS = {

    'ACTV': 1,

    'ULQD': 2,

    'LIQD': 3,

    'AMAL': 4,

    'DISD': 5,

    'NAEF': 6,

    'UPSO': 7,

    'STOF': 8,

    'D455': 9,

    'CLLP': 10,

    'CLLD': 11,

}

df\_dupfree['COMPANY\_STATUS'] = df\_dupfree['COMPANY\_STATUS'].map(mappingS)

label\_encoders = {}

categorical\_columns = ['COMPANY\_CLASS', 'COMPANY\_CATEGORY','PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN']

for column in categorical\_columns: label\_encoders[column] = LabelEncoder()

df\_dupfree[column] = label\_encoders[column].fit\_transform(df\_dupfree[column])

label\_encoder\_y = LabelEncoder()

df\_dupfree['COMPANY\_STATUS'] = label\_encoder\_y.fit\_transform(df\_dupfree['COMPANY\_STATUS'])

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

y = df\_dupfree['COMPANY\_STATUS']

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

model = RandomForestClassifier()

model.fit(X\_train, y\_train)

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

y\_pred\_decoded = label\_encoder\_y.inverse\_transform(y\_pred)

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

print(f"Accuracy: {accuracy}")

class\_labels = label\_encoder\_y.classes\_

y\_pred\_decoded = [reverse\_mapping[label] for label in y\_pred]

report = classification\_report(y\_test, y\_pred\_decoded, labels=class\_labels, zero\_division=1)

print("Classification Report:\n", report)

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

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

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues',xticklabels=label\_encoder\_y.classes\_,yticklabels=label\_encoder\_y.classes\_)

plt.xlabel('Predicted')

plt.ylabel('True')

plt.title('Confusion Matrix')

plt.show()

Random Forest is an ensemble learning method, which means it combines the predictions of multiple individual models (decision trees) to make more accurate and robust predictions. It is widely used in various domains, including business and finance, for tasks such as classification, regression, and data exploration.

Here’s how the random forest algorithm works,

* Ensemble of Decision Trees
* Bootstrap Sampling
* Feature Randomness
* Voting or Averageness

This demonstrates the steps to prepare data, build a Random Forest classifier, and evaluate its performance in predicting the status of companies based on various features. The model's accuracy and classification report provide insights into how well it performs in classifying companies into different status categories. The confusion matrix visualization helps assess the model's performance further.

