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

**Outline of project**

**Problem Statement**

The goal of this project is to develop a machine learning model that can predict the "COMPANY\_STATUS" of businesses in Tamil Nadu based on various features. The "COMPANY\_STATUS" represents the status of a company, whether it's active, dissolved, or any other status. This classification task can be valuable for various business and economic analyses.

**Design Thinking Process**

1. **Problem Definition**: Identifying the need for a predictive model to determine the status of companies in Tamil Nadu.
2. **Data Collection**: Obtaining a dataset (Data\_Gov\_Tamil\_Nadu.csv) containing relevant information about companies.
3. **Data Preprocessing**: Cleaning the dataset by handling missing values, duplicates, and outliers.
4. **Feature Engineering**: Selecting relevant features, including one-hot encoding for categorical data.
5. **Model Selection**: Choosing the Random Forest algorithm for classification.
6. **Model Training**: Training the Random Forest classifier on the preprocessed data.
7. **Model Evaluation**: Assessing the model's performance using various metrics such as accuracy, precision, recall, and F1 score.

**Phases of Development**

**1. Dataset Description**

The dataset used in this project is 'Data\_Gov\_Tamil\_Nadu.csv,' which contains information about companies in Tamil Nadu. It includes both numeric and categorical features.

**2. Data Preprocessing**

* Handling Missing Values: Rows with missing values are removed from the dataset to ensure data integrity.
* Eliminating Duplicates: Duplicate rows are removed to avoid redundancy.
* Outlier Detection: Z-scores are calculated for numeric columns, and rows with z-scores exceeding 3 in any column are removed.

**3. AI Algorithms**

The Random Forest algorithm is chosen for classification. It is an ensemble learning method that combines multiple decision trees to make predictions. Random Forest is robust, handles feature importance well, and is less prone to overfitting.

**4. Exploratory Data Analysis**

Exploratory Data Analysis (EDA) is conducted to gain insights into the dataset's characteristics, including data distribution and feature correlations. EDA helps in understanding the dataset and its potential impact on model performance.

**5. Model Performance Analysis**

The model's performance is assessed using various metrics:

* **Accuracy**: Measures the overall correctness of the model's predictions.
* **Precision**: Evaluates the accuracy of positive predictions.
* **Recall**: Gauges the model's ability to capture actual positive instances.
* **F1 Score**: Represents a balanced view of precision and recall.
* **Confusion Matrix**: Provides a detailed breakdown of true positives, true negatives, false positives, and false negatives.

**Insights**

* The Random Forest model achieved an accuracy of [insert accuracy value] on the test data.
* The weighted precision is [insert precision value], indicating that the model is [insert precision interpretation].
* The weighted recall is [insert recall value], signifying that the model captures [insert recall interpretation] of actual positive instances.
* The weighted F1 score is [insert F1 score value], demonstrating a balanced performance with respect to precision and recall.

The confusion matrix provides further insights into the model's strengths and weaknesses, allowing for a detailed analysis of true positives, true negatives, false positives, and false negatives.

This documentation serves as a comprehensive guide to the machine learning model's development, from problem definition to performance evaluation. It outlines the methodology and key results obtained from the Random Forest-based model for predicting company status in Tamil Nadu.

Top of Form

**1. Select a dataset for analysis and describe it**

1.1 About the data

In this analysis we will use the dataset "Indian Companies Registration Data [1857 - 2020]" found on [Kaggle](https://jovian.com/outlink?url=https%3A%2F%2Fwww.kaggle.com%2Frowhitswami%2Fall-indian-companies-registration-data-1900-2019).

The registered\_companies.csv file contains the details of all Indian companies ever registered in India from 1857 to 2020.

* no. of rows - 1992170
* no. of columns - 17

**Column description**:

1. CORPORATE\_IDENTIFICATION\_NUMBER - Corporate Identification Number sometimes referred to as CIN is a unique identification number which is assigned by the ROC (Registrar of Companies) of various states under the MCA (Ministry of Corporate Affairs).
2. COMPANY\_NAME - Name of the company.
3. COMPANY\_STATUS - The 'Status' tell the current state of the company. Whether it is active and operating or dormant or it has been struck off and closed. There are 13 such status that a company could be carrying.

ACTV - Active  
NAEF - Not available for e-filing  
ULQD - Under liquidation  
AMAL - Amalgamated  
STOF - Strike off  
DISD - Dissolved  
CLLD - Converted to LLP and Dissolved  
UPSO - Under process of Striking Off  
CLLP - Converted to LLP  
LIQD - Liquidated  
DRMT - Dormant  
MLIQ - Vanished  
D455 - Dormant under section 455

1. COMPANY\_CLASS - Companies are primarily classified into private and public. Private companies or private limited companies are those companies that are closely-held and have less than 200 shareholders. Public companies are limited companies that have more than 200 shareholders and are listed on a stock exchange.

Public  
Private  
Private (One Person Company)

1. COMPANY\_CATEGORY - The category of the company.

Company limited by Shares  
Company Limited by Guarantee  
Unlimited Company

1. COMPANY\_SUB\_CATEGORY - The sub-category of the company.

Non-govt company  
State Govt company  
Subsidiary of Foreign Company  
Guarantee and Association comp  
Union Govt company

1. DATE\_OF\_REGISTRATION - Date of registration of the company.
2. REGISTERED\_STATE - State in which company was registered.
3. AUTHORIZED\_CAP - Authorized Capital of the company (INR)
4. PAIDUP\_CAPITAL - Paid Up Capital of the company (INR).
5. INDUSTRIAL\_CLASS - Industrial class of the company as per NIC 2004.
6. PRINCIPAL\_ BUSINESS\_ACTIVITY\_AS\_PER\_CIN - Principal Business Activity of the company as per CIN.
7. REGISTERED\_OFFICE\_ADDRESS - Registered office address of the company.
8. REGISTRAR\_OF\_COMPANIES - Registrar office of the company.
9. EMAIL\_ADDR - Email address of the companies owner/director.
10. LATEST\_YEAR\_ANNUAL\_RETURN - Annual return of the last year.
11. LATEST\_YEAR\_FINANCIAL\_STATEMENT - Financial Statement of the last year.

1.2 Install packages and import libraries

*# Import python data analysis libraries*

import pandas as pd

import numpy as np

*# Import library to download data from Kaggle*

*# Import visualization libraries*

import matplotlib

from matplotlib import pyplot as plt

import seaborn as sns

import plotly.express as px

from plotly import graph\_objects as go

**2. Data Preprocessing and Cleaning with Pandas**

2.1 Load the dataset into a data frame using Pandas

Let's load the Indian registered companies data into a Pandas dataframe, and track the amount of time it takes using the %%time Jupyter magic command.

registered\_companies\_csv = 'all-indian-companies-registration-data-1900-2019/registered\_companies.csv'

%%time

reg\_companies\_df = pd.read\_csv(registered\_companies\_csv)

<string>:2: DtypeWarning: Columns (10) have mixed types.Specify dtype option on import or set low\_memory=False.

CPU times: user 10.2 s, sys: 918 ms, total: 11.1 s Wall time: 11.1 s

reg\_companies\_df

The dataset contains over 1900000 companies with 16 features about each company.

2.2 Explore the data

Let's get the concise summary of our dataset.

reg\_companies\_df.info(null\_counts=True)

<class 'pandas.core.frame.DataFrame'> RangeIndex: 1992170 entries, 0 to 1992169 Data columns (total 17 columns): # Column Non-Null Count Dtype --- ------ -------------- ----- 0 CORPORATE\_IDENTIFICATION\_NUMBER 1992170 non-null object 1 COMPANY\_NAME 1992170 non-null object 2 COMPANY\_STATUS 1992170 non-null object 3 COMPANY\_CLASS 1987092 non-null object 4 COMPANY\_CATEGORY 1987085 non-null object 5 COMPANY\_SUB\_CATEGORY 1987080 non-null object 6 DATE\_OF\_REGISTRATION 1989645 non-null object 7 REGISTERED\_STATE 1992170 non-null object 8 AUTHORIZED\_CAP 1992170 non-null float64 9 PAIDUP\_CAPITAL 1992170 non-null float64 10 INDUSTRIAL\_CLASS 1987359 non-null object 11 PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN 1992158 non-null object 12 REGISTERED\_OFFICE\_ADDRESS 1976911 non-null object 13 REGISTRAR\_OF\_COMPANIES 1949972 non-null object 14 EMAIL\_ADDR 1621962 non-null object 15 LATEST\_YEAR\_ANNUAL\_RETURN 1160853 non-null object 16 LATEST\_YEAR\_FINANCIAL\_STATEMENT 1163341 non-null object dtypes: float64(2), object(15) memory usage: 258.4+ MB

Most columns have the data type object, either because they contain values of different types or contain empty values (NaN). It appears that some columns contain some empty values since the *non-null* count for them is lower than the total number of rows (1992170).

Let's now view some basic statistics about numeric columns.

reg\_companies\_df.describe()

Let's extract a copy of the data from these columns into a new data frame companies\_df. We can continue to modify further without affecting the original data frame.

Let's select a subset of columns with the relevant data for our analysis.

selected\_columns = ['CORPORATE\_IDENTIFICATION\_NUMBER',

'COMPANY\_NAME',

'COMPANY\_STATUS',

'COMPANY\_CLASS',

'COMPANY\_CATEGORY',

'COMPANY\_SUB\_CATEGORY',

'DATE\_OF\_REGISTRATION',

'REGISTERED\_STATE',

'AUTHORIZED\_CAP',

'PAIDUP\_CAPITAL',

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

'REGISTERED\_OFFICE\_ADDRESS',

'LATEST\_YEAR\_ANNUAL\_RETURN',

'LATEST\_YEAR\_FINANCIAL\_STATEMENT']

companies\_df = reg\_companies\_df[selected\_columns].copy()

Let's get a summary of the dataframe companies\_df:

companies\_df.info(null\_counts=True)

From the above summary we see that there are many **missing values** in the columns LATEST\_YEAR\_ANNUAL\_RETURN and LATEST\_YEAR\_FINANCIAL\_STATEMENT. There are also a few noticeable missing values in the columns COMPANY\_CLASS, COMPANY\_CATEGORY, COMPANY\_SUB\_CATEGORY, DATE\_OF\_REGISTRATION, PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN and REGISTERED\_OFFICE\_ADDRESS.

Let us see the number of missing values in each column.

print(f"Total number of entries in the dataset : {len(companies\_df)}\n")

def **columnwise\_missing\_values**(df):

for x in df.columns:

print(f'{len(df)-df[x].count()} missing values in {x}')

columnwise\_missing\_values(companies\_df)

Total number of entries in the dataset : 1992170 0 missing values in CORPORATE\_IDENTIFICATION\_NUMBER 0 missing values in COMPANY\_NAME 0 missing values in COMPANY\_STATUS 5078 missing values in COMPANY\_CLASS 5085 missing values in COMPANY\_CATEGORY 5090 missing values in COMPANY\_SUB\_CATEGORY 2525 missing values in DATE\_OF\_REGISTRATION 0 missing values in REGISTERED\_STATE 0 missing values in AUTHORIZED\_CAP 0 missing values in PAIDUP\_CAPITAL 12 missing values in PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN 15259 missing values in REGISTERED\_OFFICE\_ADDRESS 831317 missing values in LATEST\_YEAR\_ANNUAL\_RETURN 828829 missing values in LATEST\_YEAR\_FINANCIAL\_STATEMENT

**Find and replace missing values**

*# drop rows with null values in column COMPANY\_SUB\_CATEGORY*

companies\_df = companies\_df.dropna(subset=['COMPANY\_SUB\_CATEGORY'])

print(len(companies\_df))

companies\_df.head(3)

1987080

*# check missing values in the columns*

columnwise\_missing\_values(companies\_df)

0 missing values in CORPORATE\_IDENTIFICATION\_NUMBER 0 missing values in COMPANY\_NAME 0 missing values in COMPANY\_STATUS 0 missing values in COMPANY\_CLASS 0 missing values in COMPANY\_CATEGORY 0 missing values in COMPANY\_SUB\_CATEGORY 1673 missing values in DATE\_OF\_REGISTRATION 0 missing values in REGISTERED\_STATE 0 missing values in AUTHORIZED\_CAP 0 missing values in PAIDUP\_CAPITAL 12 missing values in PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN 15233 missing values in REGISTERED\_OFFICE\_ADDRESS 826234 missing values in LATEST\_YEAR\_ANNUAL\_RETURN 823748 missing values in LATEST\_YEAR\_FINANCIAL\_STATEMENT

We see that most of the columns we are interested in now no more have null values.

companies\_df

**3. Exploratory data analysis**

**Explore columns**

Now let us explore the values in each column of companies\_df. Use the functions like max, min, unique, nunique, value\_counts, isna etc. to survey the range and distribution of values in the different columns.

*# Get a series object containing the count of unique elements in each column of dataframe*

unique\_values = companies\_df.nunique()

print('Count of unique values in each column :', )

print(unique\_values)

Count of unique values in each column : CORPORATE\_IDENTIFICATION\_NUMBER 1987080 COMPANY\_NAME 1981229 COMPANY\_STATUS 12 COMPANY\_CLASS 3 COMPANY\_CATEGORY 3 COMPANY\_SUB\_CATEGORY 5 DATE\_OF\_REGISTRATION 27648 REGISTERED\_STATE 36 AUTHORIZED\_CAP 9076 PAIDUP\_CAPITAL 144243 PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN 17 REGISTERED\_OFFICE\_ADDRESS 1782891 LATEST\_YEAR\_ANNUAL\_RETURN 554 LATEST\_YEAR\_FINANCIAL\_STATEMENT 443 dtype: int64

Let us look at the **unique values** in columns COMPANY\_STATUS, COMPANY\_CLASS, COMPANY\_CATEGORY

*# Get a series of unique values in column 'COMPANY\_STATUS' of the dataframe*

unique\_company\_status = companies\_df['COMPANY\_STATUS'].unique()

print('Unique elements in column "COMPANY\_STATUS":', '\n')

print(unique\_company\_status)

Unique elements in column "COMPANY\_STATUS": ['ACTV' 'ULQD' 'AMAL' 'STOF' 'DISD' 'NAEF' 'CLLD' 'UPSO' 'CLLP' 'D455' 'LIQD' 'DRMT']

*# Get a series of unique values in column 'COMPANY\_CLASS' of the dataframe*

unique\_company\_class = companies\_df['COMPANY\_CLASS'].unique()

print('Unique elements in column "COMPANY\_CLASS":', '\n')

print(unique\_company\_class)

Unique elements in column "COMPANY\_CLASS": ['Public' 'Private' 'Private(One Person Company)']

*# Get a series of unique values in column 'COMPANY\_CATEGORY' of the dataframe*

unique\_company\_category = companies\_df['COMPANY\_CATEGORY'].unique()

print('Unique elements in column "COMPANY\_CATEGORY":', '\n')

print(unique\_company\_category)

Unique elements in column "COMPANY\_CATEGORY": ['Company limited by Shares' 'Company Limited by Guarantee' 'Unlimited Company']

Lets now look at the **max** and **min** values in columns AUTHORIZED\_CAP and PAIDUP\_CAPITAL.

*# Get maximum and minimum value in column 'PAIDUP\_CAPITAL' of the dataframe*

paidup\_capital\_max = companies\_df['PAIDUP\_CAPITAL'].max()

paidup\_capital\_min = companies\_df['PAIDUP\_CAPITAL'].min()

print('Maximum Paidup Capital: ', paidup\_capital\_max)

print('Minimum Paidup Capital: ', paidup\_capital\_min)

Maximum Paidup Capital: 1699613000000.0 Minimum Paidup Capital: 0.0

*# Get maximum and minimum value in column 'AUTHORIZED\_CAP' of the dataframe*

authorized\_cap\_max = companies\_df['AUTHORIZED\_CAP'].max()

authorized\_cap\_min = companies\_df['AUTHORIZED\_CAP'].min()

print('Maximum Authorized Capital: ', authorized\_cap\_max)

print('Minimum Authorized Capital: ', authorized\_cap\_min)

Maximum Authorized Capital: 1850000000000.0 Minimum Authorized Capital: 0.0

Lets now look at the **value\_counts** in the columns COMPANY\_STATUS, COMPANY\_CLASS and COMPANY\_CATEGORY

companies\_df['COMPANY\_STATUS'].value\_counts()

ACTV 1186744

STOF 688727

UPSO 41443

AMAL 24892

CLLP 13175

DISD 9755

NAEF 7752

ULQD 6454

CLLD 4874

D455 2145

LIQD 1117

DRMT 2

Name: COMPANY\_STATUS, dtype: int64

companies\_df['COMPANY\_CLASS'].value\_counts()

Private 1819255

Public 137610

Private(One Person Company) 30215

Name: COMPANY\_CLASS, dtype: int64

companies\_df['COMPANY\_CATEGORY'].value\_counts()

Company limited by Shares 1963894

Company Limited by Guarantee 22219

Unlimited Company 967

Name: COMPANY\_CATEGORY, dtype: int64

companies\_df['PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN'].value\_counts(dropna=False)

Real estate renting and business activities 679906

Manufacturing 410364

Wholesale and retail trade repair of motor vehicles motorcycles and personal and household goods 227318

Construction 162119

Financial intermediation 120686

Agriculture & allied 77741

Transport storage and communications 63734

Other community social and personal service activities 59224

Extraterritorial organizations and bodies 39869

Hotels and restaurants 38516

Health and social work 34470

Education 28522

Electricity gas and water supply 22257

Mining and quarrying 20310

Public administration and defence compulsory social security 850

Unclassified 787

Activities of private households as employers and undifferentiated production activities of private households 395

NaN 12

Name: PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN, dtype: int64

companies\_df.nunique()

CORPORATE\_IDENTIFICATION\_NUMBER 1987080

COMPANY\_NAME 1981229

COMPANY\_STATUS 12

COMPANY\_CLASS 3

COMPANY\_CATEGORY 3

COMPANY\_SUB\_CATEGORY 5

DATE\_OF\_REGISTRATION 27648

REGISTERED\_STATE 36

AUTHORIZED\_CAP 9076

PAIDUP\_CAPITAL 144243

PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN 17

REGISTERED\_OFFICE\_ADDRESS 1782891

LATEST\_YEAR\_ANNUAL\_RETURN 554

LATEST\_YEAR\_FINANCIAL\_STATEMENT 443

dtype: int64

now dive into exploring our data to find some meaningful insights.

*#Working with a smaller randomly picked sample space for efficiency and overall population testing.*

*# companies\_df = companies\_df.sample(n=10000)*

*# #For readability let us use the dark mode*

*# sns.set\_theme(context='notebook',*

*# style='darkgrid',*

*# palette='magma',*

*# font='sans-serif',*

*# font\_scale=0.6,*

*# color\_codes=True,*

*# rc=None)*

Let us now visualize the status of the companies in our dataset and where they are registered.

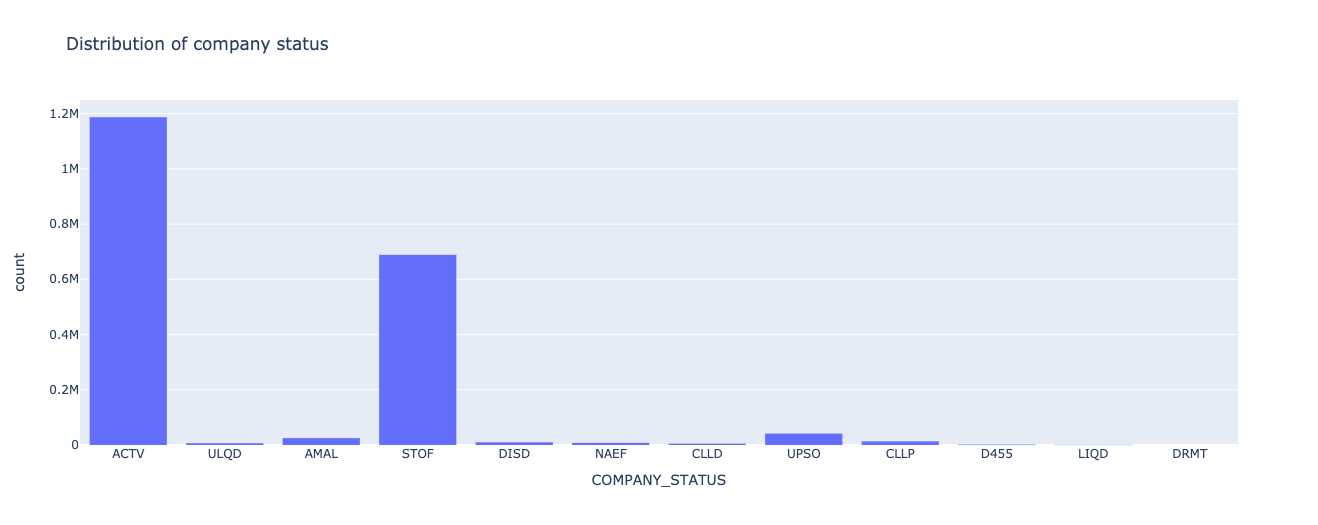
hist = px.histogram(companies\_df,

x='COMPANY\_STATUS',

title='Distribution of company status')

hist.show()

Output hidden; open in [https://colab.research.google.com](https://jovian.com/outlink?url=https%3A%2F%2Fcolab.research.google.com) to view.



As we see, out of the 1.99 million companies registered, nearly **1.2 million** are **activce(ACTV)** as of year 2020 and around **688k** companies are **Striked off (STOF)**.

hist = px.histogram(companies\_df, x='REGISTERED\_STATE')

hist.show()

Output hidden; open in [https://colab.research.google.com](https://jovian.com/outlink?url=https%3A%2F%2Fcolab.research.google.com) to view.



We see that the state of **Maharashtra** has the highest number of registration followed by **Delhi**, **West Bengal** and **Tamil Nadu**.

Interesting insights:

1. We see that West Bengal has the third highest registration of companies which was a surprise to me! Since it is the state I grew up in and people always complained about the scarcity of jobs.
2. Percentage of each company category:

*# Percentage of each category of companies as of 2020*

company\_categories = companies\_df['COMPANY\_CATEGORY'].value\_counts()

print('CATEGORIES --')

for k, c in company\_categories.items():

print(f'Percentage of {k} companies: {c/total\_companies \* 100} ')

CATEGORIES -- Percentage of Company limited by Shares companies: 98.83316222799283 Percentage of Company Limited by Guarantee companies: 1.1181734001650663 Percentage of Unlimited Company companies: 0.048664371842099964

1. Percentage of different types of companies registered:

*# Percentage of each class of companies as of 2020*

company\_classes = companies\_df['COMPANY\_CLASS'].value\_counts()

print('COMPANY CLASS --')

for k, c in company\_classes.items():

print(f'Percentage of {k} companies: {c/total\_companies \* 100} ')

COMPANY CLASS -- Percentage of Private companies: 91.55419006783823 Percentage of Public companies: 6.925237031221693 Percentage of Private(One Person Company) companies: 1.5205729009400728

*# Execute this to save new versions of the notebook*

'

**4. Ask and Answer Questions about the Data**

**Q1. Find the top 3 states with highest authorized capital and also show the status of the companies in each of the states.**

fig = px.sunburst(companies\_df,

path=['REGISTERED\_STATE', 'COMPANY\_STATUS'],

values='AUTHORIZED\_CAP',

color\_continuous\_scale='RdBu')

fig.show()

We see that Maharashtra has the highest authorized capital, followed by Delhi and then Gujarat.

**Q2. Find the number of companies registered state-wise and the top 3 among them.**

Now let's take a look at the state-wise registration of the comapnies in India.

companies\_statewise = companies\_df.groupby("REGISTERED\_STATE").size().reset\_index(name='NO\_OF\_COMPANIES')

we get a geoJSON file with all locations of states in India.

indianStates = "https://gist.githubusercontent.com/sanuann/c4acae43dead7f21976c4fe2ef4dcceb/raw/c4a3d776a05a3f49ce05747f38c86ff4cd011ee0/states-in-india.geojson"

*# Renaming the states as per GeoJSON file*

rename\_states = {"Andaman and Nicobar Islands": "Andaman & Nicobar",

"Jammu and Kashmir": "Jammu & Kashmir",

"Orissa" : "Odisha",

"Chattisgarh": "Chhattisgarh",

"Dadra and Nagra Haveli": "Dadra and Nagar Haveli and Daman and Diu",

"Pondicherry": "Puducherry",

"Uttaranchal": "Uttarakhand"}

companies\_statewise.REGISTERED\_STATE = companies\_statewise.REGISTERED\_STATE.replace(rename\_states)

Let us merge Dadra and Nagar Haveli and Daman and Diu as they are one union territory now.

daman\_diu\_companies = companies\_statewise.loc[companies\_statewise['REGISTERED\_STATE'] == "Daman and Diu"]['NO\_OF\_COMPANIES']

dadra\_nagar\_companies = companies\_statewise.loc[companies\_statewise['REGISTERED\_STATE'] == "Dadra and Nagar Haveli and Daman and Diu"]['NO\_OF\_COMPANIES']

companies\_statewise.loc[companies\_statewise['REGISTERED\_STATE'] == "Dadra and Nagar Haveli and Daman and Diu", 'NO\_OF\_COMPANIES'] = int(dadra\_nagar\_companies) + int(daman\_diu\_companies)

Let us add Ladakh to complete the map of India.

ladakh = pd.DataFrame([["Ladakh", 0.0]], columns=companies\_statewise.columns)

companies\_statewise = companies\_statewise.append(ladakh, ignore\_index=True)

fig = px.choropleth\_mapbox(

companies\_statewise,

geojson=indianStates,

featureidkey='properties.ST\_NM',

locations='REGISTERED\_STATE',

color\_continuous\_scale="YlOrBr", mapbox\_style="carto-positron", opacity=0.9,

zoom=3.4, center = {"lat": 23.473324, "lon": 78.9629},

color='NO\_OF\_COMPANIES',

title='Number of companies state-wise'

)

fig.update\_layout(margin={"r":0,"t":40,"l":0,"b":0})

fig.show()

From the above plot on the map of India,we see that the state of Maharashtra has the highest number of registration, followed by Delhi and West Bengal.

**Q3. Current status of companies**

company\_current\_status = companies\_df.groupby("COMPANY\_STATUS").size().reset\_index(name='NO\_OF\_COMPANIES')

company\_current\_status

We see that the status of the companies are defined by acronyms. Let us create a dictionary mapping the company status acronyms to its description.

*# The description of different status of companies. (Source: http://www.mca.gov.in/MinistryV2/)*

status\_description = {"ACTV": "Active",

"NAEF": "Not available for e-filing",

"ULQD": "Under liquidation",

"AMAL": "Amalgamated",

"STOF": "Strike off",

"DISD": "Dissolved",

"CLLD": "Converted to LLP and Dissolved",

"UPSO": "Under process of Striking Off",

"CLLP": "Converted to LLP",

"LIQD": "Liquidated",

"DRMT": "Dormant",

"MLIQ": "Vanished",

"D455": "Dormant under section 455"

}

Let's map the company's status to their description

company\_current\_status.COMPANY\_STATUS = company\_current\_status.COMPANY\_STATUS.replace(status\_description)

company\_current\_status

Let's visualize the above data.

fig = px.pie(company\_current\_status,

values='NO\_OF\_COMPANIES',

names='COMPANY\_STATUS',

title='Companies Current Status',

hole=.3)

fig.show()

From the above pie chart we see that nearly 60% of the companies are still active (as of year 2020). The next highest is Striked off and constitutes nearly 35%

**Q4. Find the number of company registrations over the years. Which year has the highest registration over this period?**

number\_of\_registration = companies\_df[['DATE\_OF\_REGISTRATION']].copy()

number\_of\_registration.dropna(inplace=True)

number\_of\_registration

Let us now try to extract the year of registration from the date and add the column to the number\_of\_registration dataframe.

number\_of\_registration["DATE\_OF\_REGISTRATION"] = number\_of\_registration["DATE\_OF\_REGISTRATION"].apply(pd.to\_datetime)

number\_of\_registration.head()

number\_of\_registration['YEAR\_OF\_REGISTRATION'] = number\_of\_registration['DATE\_OF\_REGISTRATION'].dt.year

number\_of\_registration['MONTH\_OF\_REGISTRATION'] = number\_of\_registration['DATE\_OF\_REGISTRATION'].dt.month

number\_of\_registration['COMPANY\_CLASS'] = companies\_df['COMPANY\_CLASS']

number\_of\_registration

Now group the number of companies registered by the year and plot it.

yearwise\_registration = number\_of\_registration.groupby("YEAR\_OF\_REGISTRATION").size().reset\_index(name='NO\_OF\_COMPANIES')

yearwise\_registration.head()

fig = px.bar(yearwise\_registration, x='YEAR\_OF\_REGISTRATION', y='NO\_OF\_COMPANIES', title='Number of registrations over the years 1857 - 2020')

fig.show()

From the above plot we see that regsitrations took off from around the year 1980. There was a steady rise and reached a peak in the year 1995 and then the registration fell off around the year 2000 - 2001. This is perhaps during the recession period. After that the number of registrations has been on a steady rise.

Over all the years we see that the maximum registrations took place in the year 2019.

**Q5. Find statewise registrations in the year 2019 and also categorize into business activity type.**

number\_of\_registration.head()

number\_of\_registration['REGISTERED\_STATE'] = companies\_df['REGISTERED\_STATE']

number\_of\_registration['PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN'] = companies\_df['PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN']

number\_of\_registration

*# companies\_2019 = companies\_df[['REGISTERED\_STATE', 'DATE\_OF\_REGISTRATION', 'PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN']].copy()*

*# companies\_2019.dropna(subset=['DATE\_OF\_REGISTRATION'], inplace=True)*

*# companies\_2019['YEAR\_OF\_REGISTRATION'] = pd.to\_datetime(companies\_2019['DATE\_OF\_REGISTRATION'], errors = 'coerce').dt.year*

*# companies\_2019['MONTH\_OF\_REGISTRATION'] = pd.to\_datetime(companies\_2019['DATE\_OF\_REGISTRATION'], errors = 'coerce').dt.month*

*# companies\_2019.head()*

*# companies\_2019.dropna(subset=['YEAR\_OF\_REGISTRATION'], inplace=True)*

*# companies\_2019.YEAR\_OF\_REGISTRATION = companies\_2019.YEAR\_OF\_REGISTRATION.astype(int)*

companies\_2019 = number\_of\_registration[number\_of\_registration['YEAR\_OF\_REGISTRATION'] == 2019]

companies\_2019

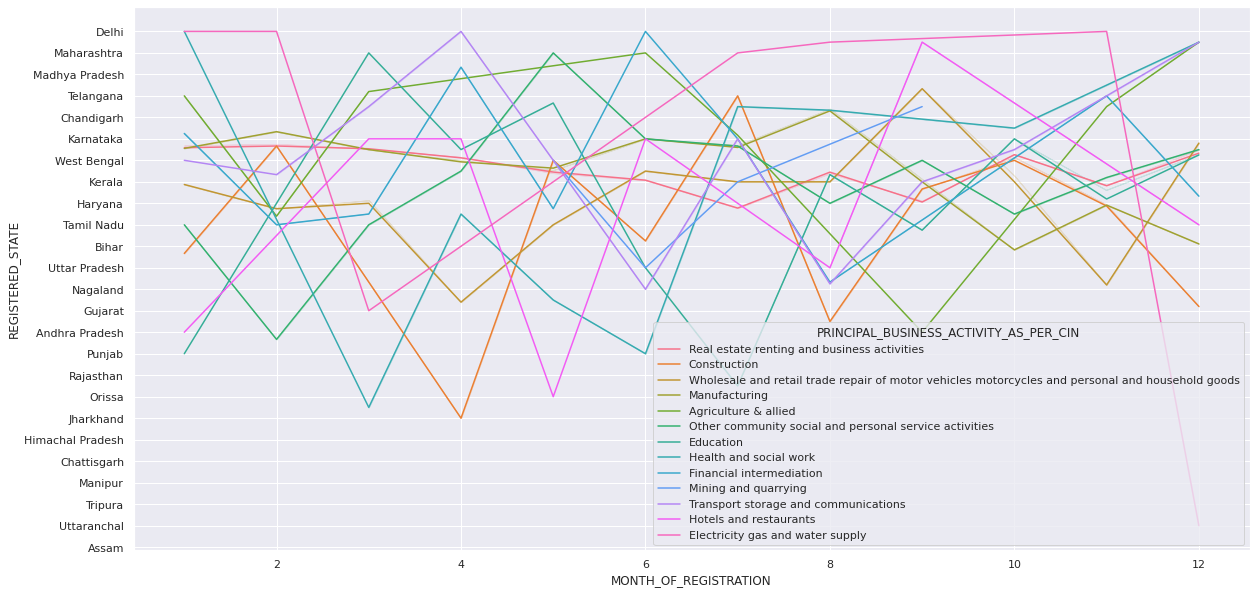
'

sns.set(rc={'figure.figsize': (20,10)})

sns.lineplot(x='MONTH\_OF\_REGISTRATION', y='REGISTERED\_STATE', data=companies\_2019, hue='PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN',

ci=False, markers=True)

<matplotlib.axes.\_subplots.AxesSubplot at 0x7fbd2cd95510>



companies\_2019.groupby('REGISTERED\_STATE')

<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7fbd2cdb04d0>

**Q6. Top 20 companies with highest Authorized Capital and its Paidup Capital (INR)**

highest\_auth\_capital\_companies = companies\_df.sort\_values(by='AUTHORIZED\_CAP', ascending=False)[0:20]

fig = go.Figure()

fig.add\_trace(go.Bar(y=highest\_auth\_capital\_companies['COMPANY\_NAME'],

x=highest\_auth\_capital\_companies['AUTHORIZED\_CAP'],

text=highest\_auth\_capital\_companies['AUTHORIZED\_CAP'],

name="Authorized Capital (INR)",

orientation='h'))

fig.add\_trace(go.Bar(y=highest\_auth\_capital\_companies['COMPANY\_NAME'],

x=highest\_auth\_capital\_companies['PAIDUP\_CAPITAL'],

text=highest\_auth\_capital\_companies['PAIDUP\_CAPITAL'],

name="Paid Up Capital (INR)",

orientation='h'))

fig.update\_layout(

autosize=False,

width=1300,

height=1000,

barmode='group',

bargap=0.2,

font=dict(size=10))

fig.update\_traces(textposition='outside')

fig.show()

We see that "Jio Platforms" has the highest authorized capital as well as Paidup Capital whereas the lowest authorized capital is for "Bharat Sanchar Nigam Limited".

**Q7. Find the highest Principal Business Activity of a company as per CIN**

company\_business\_activity = companies\_df.groupby("PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN").size().reset\_index(name='NO\_OF\_COMPANIES')

fig = px.pie(company\_business\_activity, values='NO\_OF\_COMPANIES', names='PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN', title='Principal Business Activity of a company as per CIN (Hover to see the data)', hole=.3)

fig.update\_layout(showlegend=False)

fig.update\_traces(textposition='outside', textinfo='percent')

fig.show()

**Q8. Yearwise Principal Business activity for the last 10 years (2010-2020)**

col\_list = ["COMPANY\_NAME", "COMPANY\_STATUS","DATE\_OF\_REGISTRATION", "REGISTERED\_STATE", "EMAIL\_ADDR", "PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN"]

df = pd.read\_csv(registered\_companies\_csv, usecols=col\_list)

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

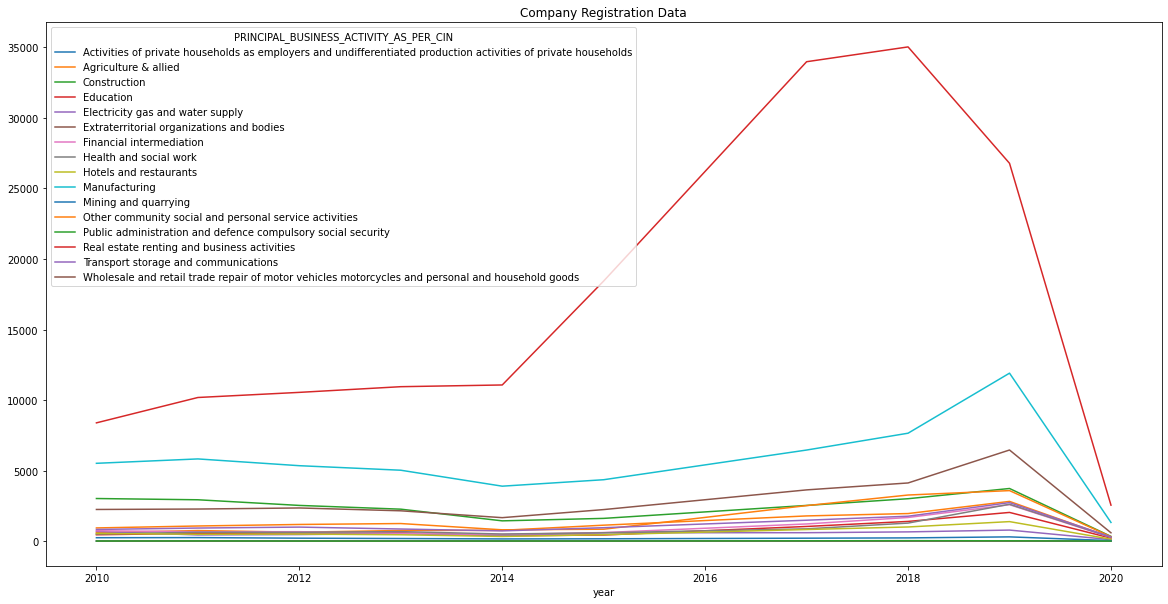
search = df.loc [df['REGISTERED\_STATE'].isin(states) & df['year'].isin(years) & df['COMPANY\_STATUS'].str.contains("ACTV",case=False)]

search.head()

table = search.groupby('year')['PRINCIPAL\_BUSINESS\_ACTIVITY\_AS\_PER\_CIN'].value\_counts().unstack().fillna(0)

table

plotting = table.plot.line(figsize=(20,10), fontsize=10, title="Company Registration Data", legend=True)



From the above plot we can see that Education sector had the highest registration during the last 10 years. It was highest in 2018 and then declined. The next highest registration was in the manufacturing sector with its highest in 2019.