AI-DRIVEN EXPLORATION AND PREDICTION OF COMPANY REGISTRATION TRENDS WITH REGISTER OF COMPANIES (ROC) IN DATA PRE PROCESSING

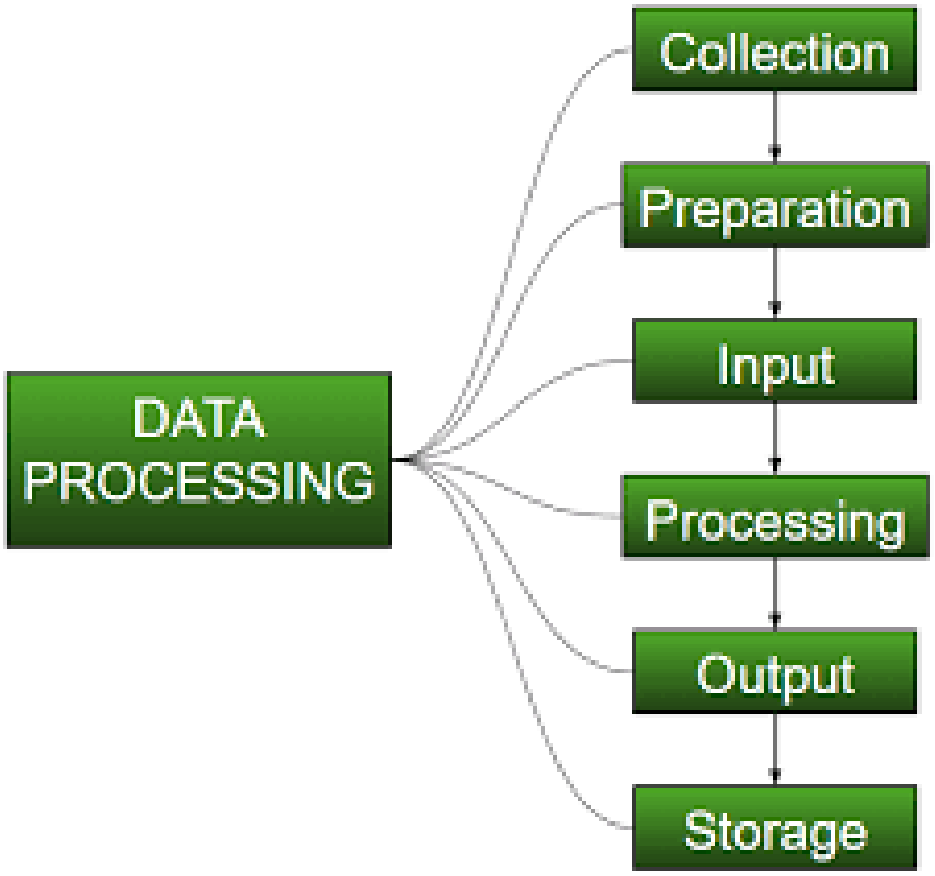
PHASE-2

DATA COLLECTION:

Obtain access to RoC data, which may require permissions or APIs.
Collect historical company registration data , including details like company names, registration dates, locations, and industry types. Artificial intelligence (AI) coupled with promising machine learning (ML) techniques well known from computer science is broadly affecting many aspects of various fields including science and technology, industry, and even our day-to-day life. The ML techniques have been developed to analyze high-throughput data with a view to obtaining useful insights, categorizing, predicting, and making evidence-based decisions in novel ways which will promote the growth of novel applications and fuel the sustainable booming of AI.

DATA PREPROCESSING:

Clean the data by handlingmissingvalues,duplicates,andoutliersconverttext-basedinformation into numerical features using techniques like one-hot encoding or word embedding. Registrars of Companies (ROC) appointed under Section 609 of the Companies Act covering the various States and Union Territories are vested with the primary duty of registering companies and LLPs floated in the respective states and the Union Territories and ensuring that such companies and LLPs comply with statutory requirements under the Act. These offices function as registry of records, relating to the companies registered with them, which are available for inspection by members of public on payment of the prescribed fee. The Central Government exercises administrative control over these offices through the respective Regional Directors.



EXPLORATORY DATA ANALYSIS(EDA):

Perform data visualization and statistical analysis to understand trends, correlations, and patterns in the data. Identify factors that might influence company registrations, such as economic indicators or regional demographics. The main objective of this article is to cover the steps involved in Data pre-processing, Feature Engineering, and different stages of Exploratory Data Analysis, which is an essential step in any research analysis. Data pre-processing, Feature Engineering, and EDA are fundamental early steps after data collection

TOP 10 EDA COMPANIES
WORLDWIDE—2000 CALENDAR YEAR

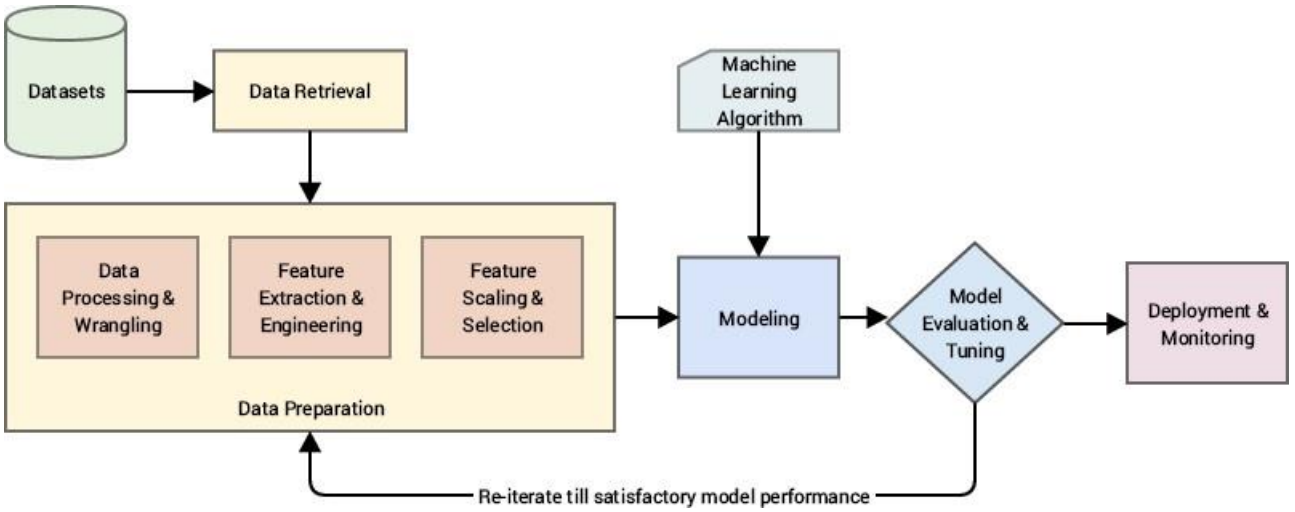
Rank	Company (Headquarters)	Worldwide EDA Sales (millions)
1	Cadence Design Systems (San Jose)	1,279.6
2	Synopsys (Mountain View, Calif.)	724.1
3	Mentor Graphics (Wilsonville, Ore.)	589.8
3	Avant (Fremont, Calif.)	358.1
5	Zuken (Yokohama, Japan) [1]	161.3
6	Innoveda (Marlborough, Mass.) [2]	89.9
7	IKOS Systems (San Jose)	78.4
8	Ansoft (Pittsburgh, Pa.) [3]	40.2
9	Synplicity (Sunnyvale, Calif)	34.6
10	Protel International (Frenchs Forest, Australia)	25.6

[1] Zuken's EDA sales figure is for fiscal year end 3/31/00.
[2] Innoveda was created by the merger of Viewlogic Systems and Summit Design on March 23, 2000.
[3] EDA sales figures represent total sales.

SOURCE: CAHNERS RESEARCH

FEATURE ENGINEERING:

Create new features or transform existing ones to capture relevant information for prediction. Consider time-based features like seasonality and trends. The use of Machine Leaning(ML)has increased substantially in enterprise data analytics scenarios to extract valuable insights from the business data. Hence, it is very important to have an ecosystem to build, test, deploy, and maintain the enterprise grade machine learning models in production environments

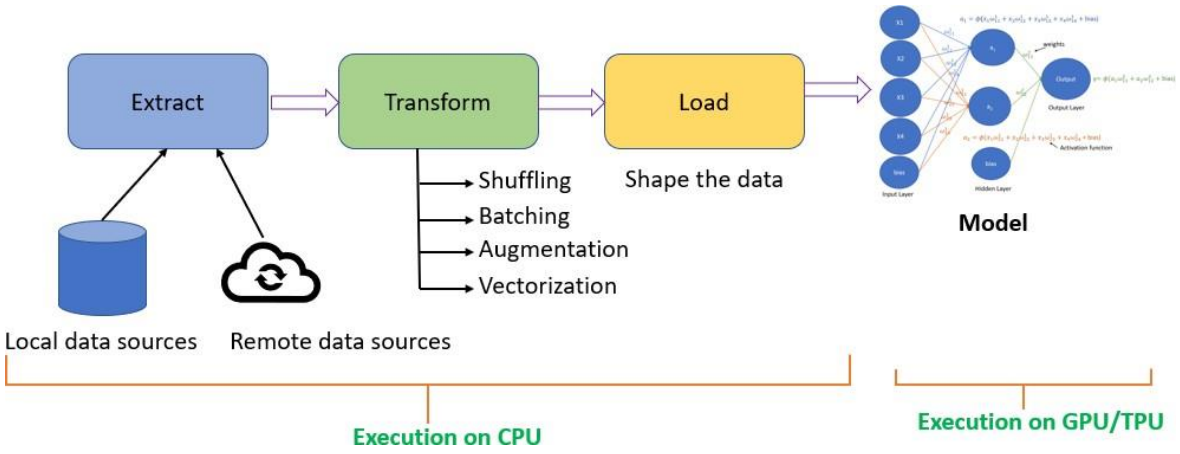


MODEL SELECTION:

Choose appropriate machine learning or deep learning models for time-series forecasting. Time series models like ARIMA or machine learning models like Random Forests and LSTM neural networks are commonly used. In this contribution, we investigate a new integrated solution of predictive model-based quality inspection in industrial manufacturing by utilizing Machine Learning techniques and Edge Cloud
Computing technology. In contrast to state-of-the-art contributions, we propose a holistic approach comprising the target-oriented data acquisition and processing, modeling and model deployment as well as the technological implementation in the existing IT plant infrastructure. A real industrial use case in SMT manufacturing is presented to underline the procedure and benefits of the proposed method. The results show that by employing the proposed method, inspection volumes can be reduced significantly and thus economic advantages can be generated.

MODEL TRAINING:

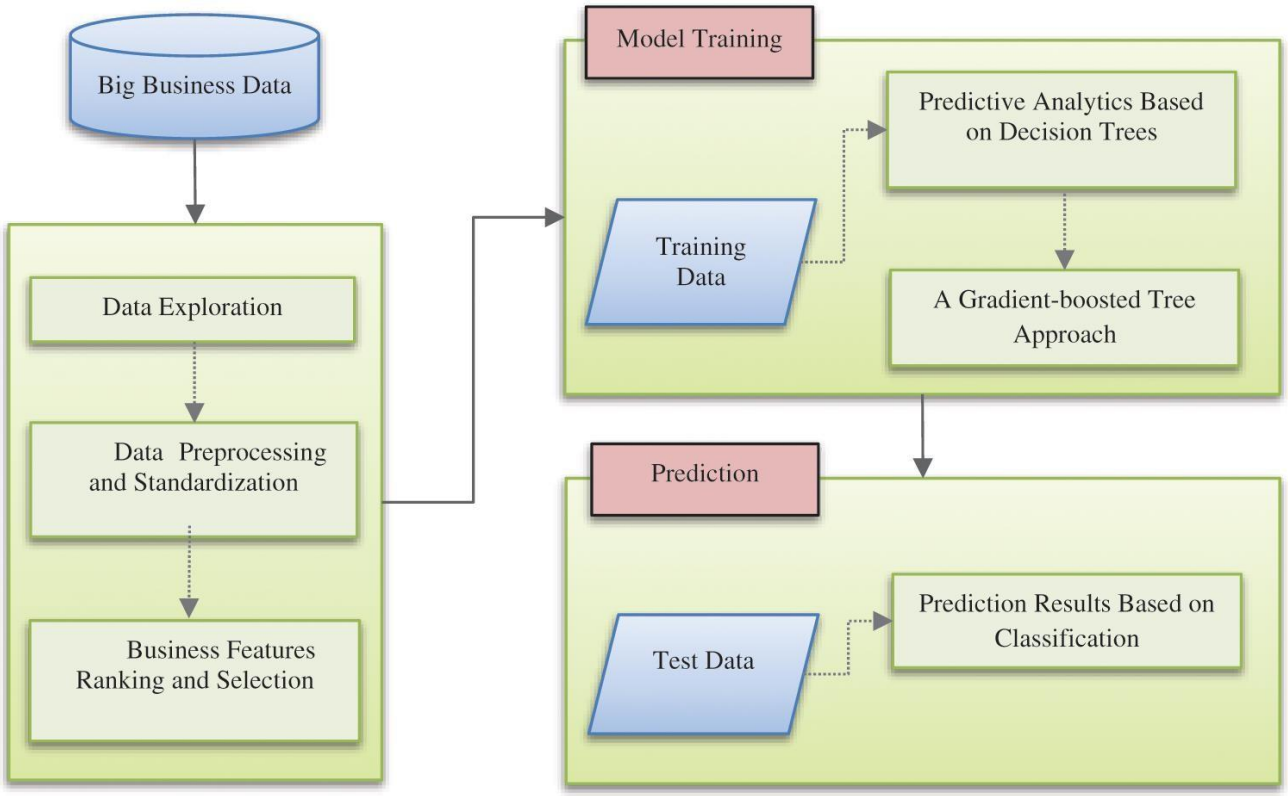
Split the data into training and testing sets. Train your selected models on historical data, using various hyper parameters and optimization techniques.



MODEL EVALUATION:

Evaluate model performance using appropriate metrics like Mean Absolute Error(MAE), Mean Squared Error(MSE),or Root Mean Squared Error (RMSE). Even though many of the contributions to AI systems originate from computer science, statistics has played an important role throughout. Early examples occurred in the context of realizing the relationship between back propagation and nonlinear least squares methods, see, e.g., Warner and Misra (1996). Important ML methods such as random forests (Breiman 2001) or support vector machines (Cortes and Vapnik 1995) were developed by statisticians. Others, like radial basis function networks (Chen et al. 1991), can alsobeconsideredandstudiedasnonlinearregressionmodelsinstatistics.

applications based on AI algorithms have played a significant role in various fields and subjects, on the basis of which the prosperity of the DL framework and platform has been founded. AI frameworks and platforms reduce the requirement of accessing AI technology by integrating the overall process of algorithm development, which enables researchers from different areas to use it across other fields, allowing them to focus on designing the structure of neural networks, thus providing better solutions to problems in their fields. At the beginning of the 21st century ,only a few tools, such as MATLAB, Open NN, and Torch, were capable of describing and developing neural networks. However, these tools were not originally designed for AI models, and thus faced problems, such as complicated user API and lacking GPU support



DEPLOYMENT:

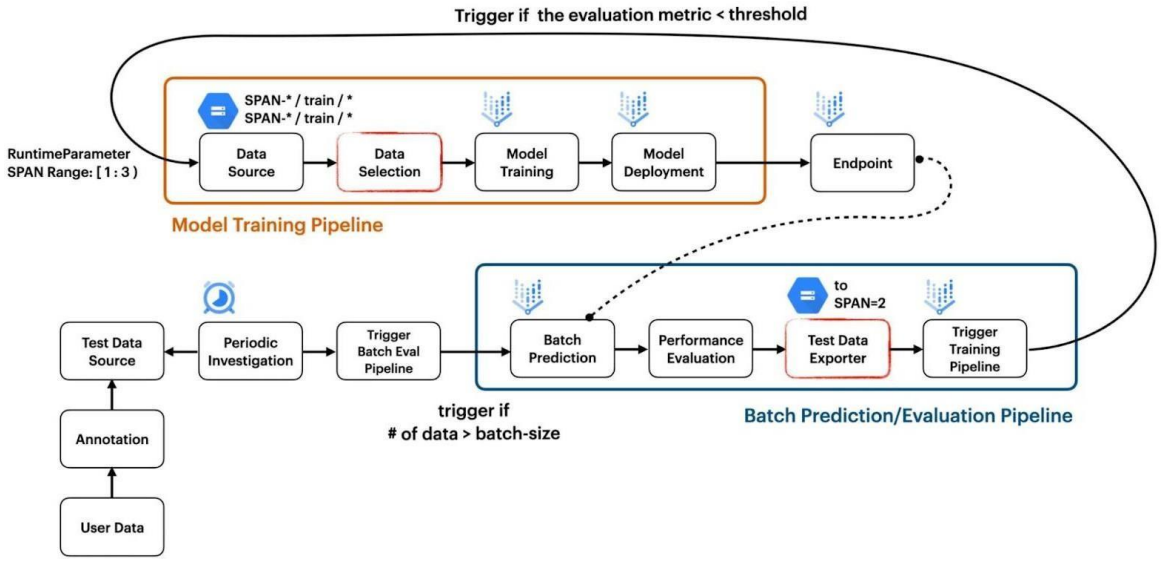
Develop a user-friendly interface for stakeholders to interact with the AI-driven system.

Deploy the system and ensure it can handle real-time or batch predictions. In this stage, you focus on building and training the AI model. Select appropriate machine learning algorithms, frameworks, and tools. Train and fine-tune our model using the chosen algorithm, considering factors like hyper parameters and architecture. This stage may Involve experimenting with different approaches to improve model performance.

MONITORINGANDMAINTENANCE:

Continuously monitor the model's performance and retrain it with new data as it becomes available.

Update the system as needed to adapt to changing trends or regulations Utilizing AI for predictive maintenance can bring about many advantages for facilities engineering; for instance, it can enhance reliability and availability of assets by avoiding breakdowns, delays, and disruptions. Additionally, it can improve safety and quality by minimizing human errors, accidents, and defects. Moreover, AI can help save costs and resources by optimizing maintenance schedules, reducing waste, and prolonging equipment life span. Furthermore, it can boost productivity and performance by increasing efficiency, output, and customer satisfaction. Lastly, AI can support innovation and sustainability by allowing data-driven decision making, continuous improvement, and environmental compliance.



Program :

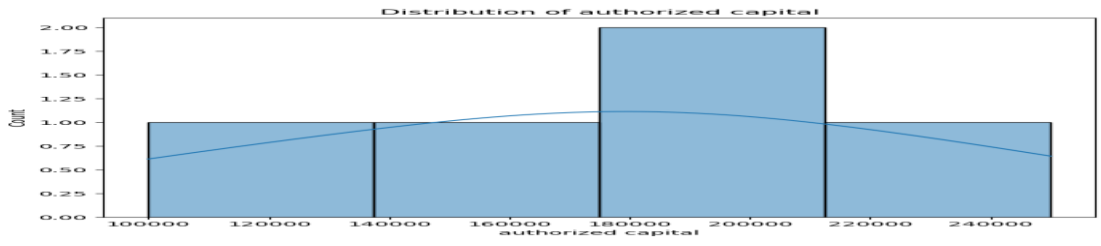
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_curve, roc_auc_score, auc
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.metrics import accuracy_score, precision_score, recall_sc
ore, f1_score
from sklearn.metrics import mean_absolute_error, mean_squared_error,
  r2_score
from statsmodels.tsa.statespace.sarimax import SARIMAX
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
```

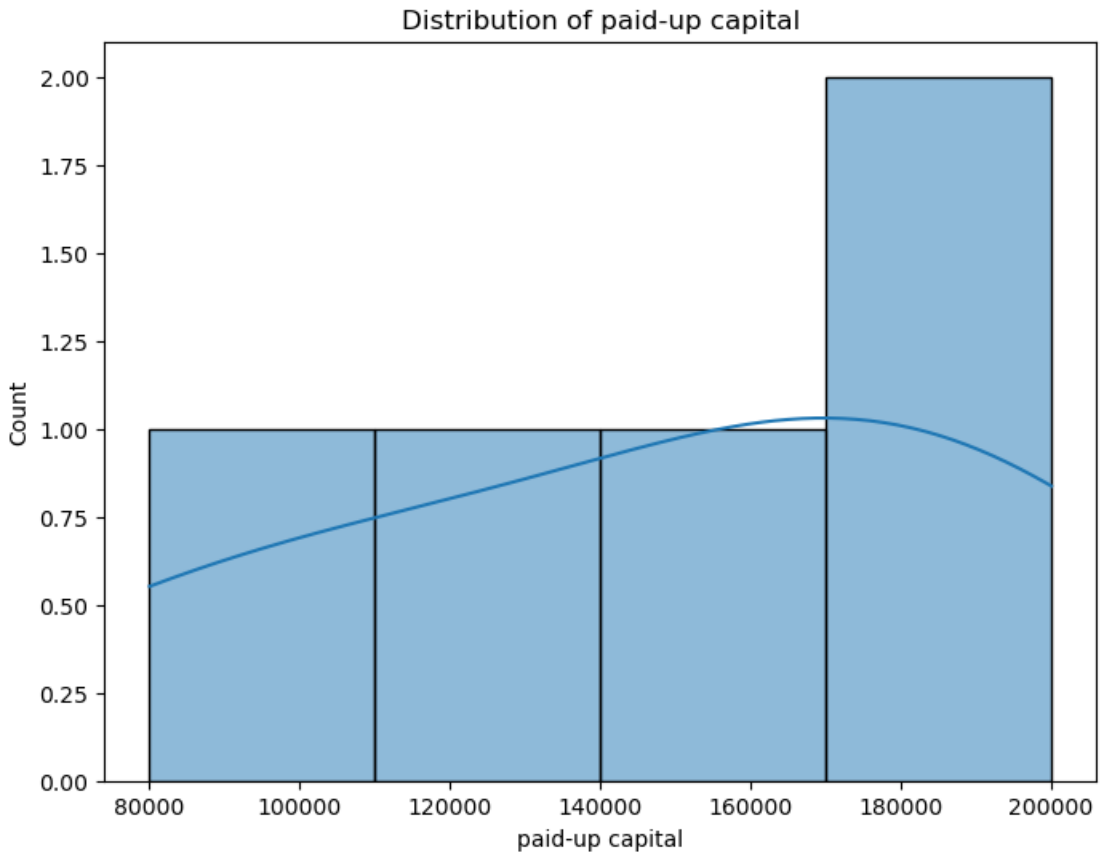
```
data=pd.read_csv('Data_Gov_Tamil_Nadu.csv',encoding='ISO-8859-1')

data.dropna(inplace=True)

print(data.columns)
```

Index(['CORPORATE_IDENTIFICATION_NUMBER', 'COMPANY_NAME', 'COMP
ANY_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'] dtype='object')

```
data = pd.DataFrame({
"company name": ["CompanyA", "CompanyB", "CompanyC"],
"status": ["Active", "Inactive", "Active"],
"class": ["Class1", "Class2", "Class1"],
"category": ["CategoryX", "CategoryY", "CategoryZ"]
})

label_encoder = LabelEncoder()
categorical_columns = ["company name", "status", "class", "category"]
for col in categorical_columns:
    data[col] = label_encoder.fit_transform(data[col])

data = pd.DataFrame({
"authorized capital": [100000, 200000, 150000, 250000, 180000],
"paid-up capital": [80000, 180000, 120000, 200000, 160000]
})

numeric_columns = ["authorized capital", "paid-up capital"]

# Loop through numeric columns and create histogram plots
for col in numeric_columns:
    plt.figure(figsize=(8, 6))
    sns.histplot(data=data, x=col, kde=True)
    plt.title(f'Distribution of {col}')
    plt.show()
```



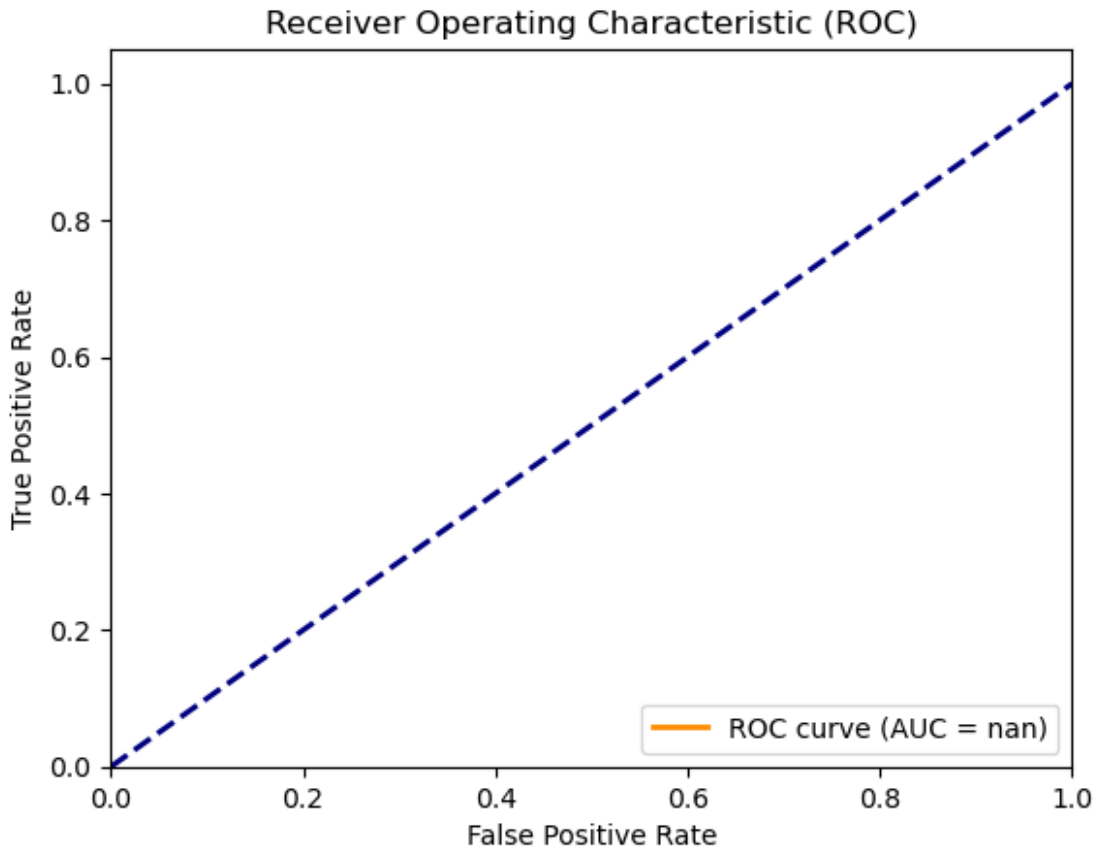



```
print(data.columns)
```

Index(['authorized capital', 'paid-up capital'], dtype='object')

```
train_size =int(len(data) *0.8)
train_data = data[:train_size]
test_data = data[train_size:]

X = data.drop("authorized capital", axis=1)  # Drop 'authorized capita
l' from features
y = data["paid-up capital"]
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.
2, random_state=42)

model = RandomForestClassifier()  # You can use a different algorithm
as per your requirement
model.fit(X_train, y_train)
y_pred = model.predict(X_test)

positive_class_label =180000
fpr, tpr, thresholds = roc_curve(y_test, y_pred, pos_label=positive_cl
ass_label)
roc_auc = auc(fpr, tpr
```

```
plt.figure()
plt.plot(fpr, tpr, color='darkorange', lw=2, label=f'ROC curve (AUC =
{roc_auc:.2f})')
plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')
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()
```




```
model=Sequential()
model.add(LSTM(units=50, activation='relu', input_shape=(X_train.shape
[1], 1)))
model.add(Dense(units=1))
model.compile(optimizer='adam', loss='mean_squared_error')

model.fit(X_train, y_train, epochs=50, batch_size=32)
```

```
Epoch 1/50
1/1 [==============================] - 1s 894ms/step - loss: 23613347840.0000
```

```
Epoch 2/50
1/1 [==============================] - 0s 3ms/step - loss: 23537295360.0000
Epoch 3/50
1/1 [==============================] - 0s 3ms/step - loss: 23461740544.0000
Epoch 4/50
1/1 [==============================] - 0s 0s/step - loss: 23386685440.0000
Epoch 5/50
1/1 [==============================] - 0s 0s/step - loss: 23312140288.0000
Epoch 6/50
1/1 [==============================] - 0s 0s/step - loss: 23238105088.0000
Epoch 7/50
1/1 [==============================] - 0s 16ms/step - loss: 23164583936.0000
Epoch 8/50
1/1 [==============================] - 0s 0s/step - loss: 23091580928.0000
Epoch 9/50
1/1 [==============================] - 0s 0s/step - loss: 23019094016.0000
Epoch 10/50
1/1 [==============================] - 0s 0s/step - loss: 22947110912.0000
Epoch 11/50
1/1 [==============================] - 0s 15ms/step - loss: 22885529600.0000
Epoch 12/50
1/1 [==============================] - 0s 3ms/step - loss: 22825199616.0000
Epoch 13/50
1/1 [==============================] - 0s 3ms/step - loss: 22765271040.0000
Epoch 14/50
1/1 [==============================] - 0s 3ms/step - loss: 22705743872.0000
Epoch 15/50
1/1 [==============================] - 0s 3ms/step - loss: 22646614016.0000
Epoch 16/50
1/1 [==============================] - 0s 0s/step - loss: 22587885568.0000
Epoch 17/50
1/1 [==============================] - 0s 0s/step - loss: 22529556480.0000
Epoch 18/50
1/1 [==============================] - 0s 0s/step - loss: 22471620608.0000
Epoch 19/50
1/1 [==============================] - 0s 15ms/step - loss: 22414080000.0000
Epoch 20/50
1/1 [==============================] - 0s 871us/step - loss: 22356930560.0000
Epoch 21/50
1/1 [==============================] - 0s 0s/step - loss: 22300168192.0000
Epoch 22/50
1/1 [==============================] - 0s 0s/step - loss: 22243790848.0000
Epoch 23/50
1/1 [==============================] - 0s 14ms/step - loss: 22187790336.0000
Epoch 24/50
1/1 [==============================] - 0s 2ms/step - loss: 22132164608.0000
```

```
Epoch 25/50
1/1 [==============================] - 0s 0s/step - loss: 22076907520.0000
Epoch 26/50
1/1 [==============================] - 0s 0s/step - loss: 22022004736.0000
Epoch 27/50
1/1 [==============================] - 0s 0s/step - loss: 21967460352.0000
Epoch 28/50
1/1 [==============================] - 0s 3ms/step - loss: 21913260032.0000
Epoch 29/50
1/1 [==============================] - 0s 0s/step - loss: 21868576768.0000
Epoch 30/50
1/1 [==============================] - 0s 0s/step - loss: 21826981888.0000
Epoch 31/50
1/1 [==============================] - 0s 0s/step - loss: 21785632768.0000
Epoch 32/50
1/1 [==============================] - 0s 15ms/step - loss: 21744523264.0000
Epoch 33/50
1/1 [==============================] - 0s 0s/step - loss: 21703649280.0000
Epoch 34/50
1/1 [==============================] - 0s 0s/step - loss: 21663012864.0000
Epoch 35/50
1/1 [==============================] - 0s 0s/step - loss: 21622603776.0000
Epoch 36/50
1/1 [==============================] - 0s 16ms/step - loss: 21582419968.0000
Epoch 37/50
1/1 [==============================] - 0s 0s/step - loss: 21542459392.0000
Epoch 38/50
1/1 [==============================] - 0s 0s/step - loss: 21502717952.0000
Epoch 39/50
1/1 [==============================] - 0s 0s/step - loss: 21463191552.0000
Epoch 40/50
1/1 [==============================] - 0s 16ms/step - loss: 21423874048.0000
Epoch 41/50
1/1 [==============================] - 0s 0s/step - loss: 21384763392.0000
Epoch 42/50
1/1 [==============================] - 0s 0s/step - loss: 21345853440.0000
Epoch 43/50
1/1 [==============================] - 0s 0s/step - loss: 21307144192.0000
Epoch 44/50
1/1 [==============================] - 0s 14ms/step - loss: 21268627456.0000
Epoch 45/50
1/1 [==============================] - 0s 0s/step - loss: 21230301184.0000
Epoch 46/50
1/1 [==============================] - 0s 0s/step - loss: 21192159232.0000
Epoch 47/50
1/1 [==============================] - 0s 0s/step - loss: 21154201600.0000
```

```
Epoch 48/50
1/1 [==============================] - 0s 15ms/step - loss: 21116420096.0000
Epoch 49/50
1/1 [==============================] - 0s 0s/step - loss: 21078808576.0000
Epoch 50/50
1/1 [==============================] - 0s 0s/step - loss: 21041364992.0000

<keras.src.callbacks.History at 0x2d60d936880>
```

```
predictions = model.predict(X_test)
```

```
1/1 [==============================] - 0s 120ms/step
```

```
mae = mean_absolute_error(y_test, predictions)

mse = mean_squared_error(y_test, predictions)

rmse = np.sqrt(mse)

print("Mean Absolute Error (MAE):", mae)
print("Mean Squared Error (MSE):", mse)
print("Root Mean Squared Error (RMSE):", rmse)
```

Mean Absolute Error (MAE): 177499.68139648438
Mean Squared Error (MSE): 31506136895.853462
Root Mean Squared Error (RMSE): 177499.68139648438