PredictiveModelTaskReport

November 16, 2023

1 Predictive Model

[1]:
```
import sys
sys.path.append("../")
```

[2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from prophet import Prophet
import plotly.offline as py
py.init_notebook_mode()

from utils.utils import *
from model.dl_models import rgu_model,plot_gru_synthesize_charts
from model.ml_models import MachineLearningModel

from data_processing.dataset_generation import DatasetGeneration
from config.variables import features_path, data_path,result_path
```

```
C:\Users\Admin\AppData\Local\Programs\Python\Python311\Lib\site-
packages\tqdm\auto.py:21: TqdmWarning: IProgress not found. Please update
jupyter and ipywidgets. See
https://ipywidgets.readthedocs.io/en/stable/user_install.html
  from .autonotebook import tqdm as notebook_tqdm
```

1.1 There are 5 main steps to mining data:

1. **Data Generation**: We generate 3 dataset such as date_smedebtsu.csv, lag3_smedebtsu.csv, lag3_date_smedebtsu.csv
2. **Utilizing Statistical Models for Forecasting**: In this step, we train the model Prophet to analyze the model's results.
3. **Executing baseline machine learning models**: In this part, we use machine learning Linear Regression, Forest Regressor, Boosting Regressor, and Decision Tree Regressor to find out the most effective models.
4. **Train forecasting models using GRU model**: We propose a model deep-learning(GRU) to predict total debts.

5. **Experimental results** We will analyze of above models over 3 datasets: 'date_smedebtsu,' 'lag3_smedebtsu,' and 'lag3_date_smedebtsu'

1.2 1. Data Generation

- Calculate the total debts from all lenders over time
- Generating a dataset with additional date features, including 'day', 'month', 'year', 'quarter', 'dayofweek', and 'dayofyear'. The resulting DataFrame is then saved as a CSV file named 'date_smedebtsu.csv'.
- Creating a dataset with lagged features like 'totalU_lag1', 'totalU_lag2', 'totalU_lag3'. The resulting DataFrame is then saved as a CSV file named 'lag3_date_smedebtsu.csv'

[3]:
```
dataset_generation = DatasetGeneration('../../data/processed/processed_smedebtsu.
↪csv')
dataset_generation.dataset_generation()
```

```
File path:  ../../data/features\date_smedebtsu.csv
File path:  ../../data/features\lag3_smedebtsu.csv
File path:  ../../data/features\lag3_date_smedebtsu.csv
```

[4]:
```
# Show the generated dataset
dataset_date_debts = pd.read_csv(os.path.join(features_path, 'date_smedebtsu.
↪csv'))
dataset_date_debts.head()
```

[4]:

	Date_time	totalU	day	month	year	quarter	dayofweek	dayofyear
0	2013-10-13	228007.01	13	10	2013	4	6	286
1	2013-11-13	227988.31	13	11	2013	4	2	317
2	2013-12-10	265199.00	10	12	2013	4	1	344
3	2014-01-23	299453.00	23	1	2014	1	3	23
4	2014-03-05	290103.00	5	3	2014	1	2	64

[5]:
```
# Show the generated dataset
dataset_lag3_debts = pd.read_csv(os.path.join(features_path, 'lag3_smedebtsu.
↪csv'))
dataset_lag3_debts.head()
```

[5]:

	Date_time	totalU	totalU_lag1	totalU_lag2	totalU_lag3
0	2014-01-23	299453.0	265199.0	227988.31	228007.01
1	2014-03-05	290103.0	299453.0	265199.00	227988.31
2	2014-04-05	304337.0	290103.0	299453.00	265199.00
3	2014-05-05	293623.0	304337.0	290103.00	299453.00
4	2014-06-06	307582.0	293623.0	304337.00	290103.00

[6]:
```
# Show the generated dataset
lag3_date_smedebtsu = pd.read_csv(os.path.join(features_path,␣
↪'lag3_date_smedebtsu.csv'))
lag3_date_smedebtsu.head()
```

[6]:

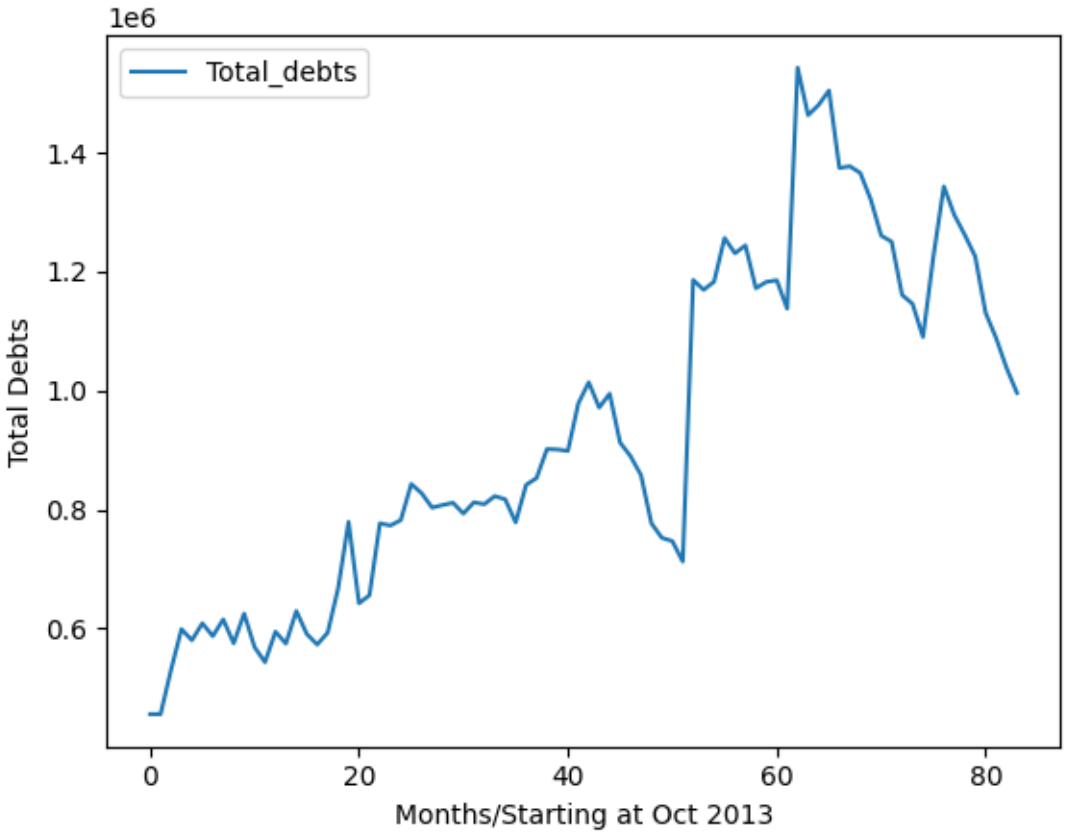
	Date_time	totalU	day	month	year	quarter	dayofweek	dayofyear	\
0	2014-01-23	299453.0	23	1	2014	1	3	23	
1	2014-03-05	290103.0	5	3	2014	1	2	64	
2	2014-04-05	304337.0	5	4	2014	2	5	95	
3	2014-05-05	293623.0	5	5	2014	2	0	125	
4	2014-06-06	307582.0	6	6	2014	2	4	157	

	totalU_lag1	totalU_lag2	totalU_lag3
0	265199.0	227988.31	228007.01
1	299453.0	265199.00	227988.31
2	290103.0	299453.00	265199.00
3	304337.0	290103.00	299453.00
4	293623.0	304337.00	290103.00

1.3 2. Utilizing Statistical Models for Forecasting

- Statistical models, such as Autoregressive (AR), Moving Average (MA),Prophet, are commonly employed for forecasting

[7]:
```
# We focus on the month of the year and the total debts. Plot a graph
df = pd.read_csv(data_path)
df['Total_debts'] = df.sum(axis=1, numeric_only=True)
df_total_debts = df[['Date_time', 'Total_debts']]
df_total_debts.plot()
plt.xlabel("Months/Starting at Oct 2013")
plt.ylabel("Total Debts")
plt.show()
```



1.3.1 Model Prophet

-**Prophet** is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects.

Prophet has several advantages associated with it. These are given below: 1. Accurate and fast - Prophet is accurate and fast. It is used in many applications across Facebook for producing reliable forecasts for planning and goal setting. 2. Robust to outliers - It is robust to outliers. It handles outliers by removing them. 3. Robust to missing data - Prophet is resilient to missing data.

[8]:
```
# Split data into train and test sets using a specific day '2021-04-25'.
# This ensures that the results can be compared with other methods using the␣
 ↪same test set

df_total_debts['Date_time'] = pd.to_datetime(df_total_debts['Date_time'])

def split_prophet(df):
    boundary_day_test = '2021-04-25'
    train_df = df[df['Date_time'] < boundary_day_test]
    test_df = df[df['Date_time'] >= boundary_day_test]
```

```
    print("--------------Total Debts-----------------")
    print(f"Train Size: {len(train_df)}, Test Size: {len(test_df)}")

    return train_df, test_df

train_df, test_df = split_prophet(df_total_debts)
```

```
--------------Total Debts-----------------
Train Size: 65, Test Size: 19
```

[9]:
```
df = train_df.rename(columns={'Date_time': 'ds',
                        'Total_debts': 'y'})
my_model = Prophet()
my_model.fit(df.reset_index())
```

```
09:49:34 - cmdstanpy - INFO - Chain [1] start processing
09:49:34 - cmdstanpy - INFO - Chain [1] done processing
```

[9]:
```
<prophet.forecaster.Prophet at 0x285918ed490>
```

[10]:
```
prophet_test_df = test_df.rename(columns={'Date_time': 'ds',
                        'Total_debts': 'y'})
forecast = my_model.predict(prophet_test_df[["ds"]])
forecast.index = pd.RangeIndex(start=65, stop=84, step=1)
prophet_prediction_df = forecast.rename(columns={'ds':'Date_time',
                        'yhat':'Total_debts'})
```

[11]:
```
prophet_prediction_df.head()
```

[11]:

	Date_time	trend	yhat_lower	yhat_upper	trend_lower	trend_upper	additive_terms	additive_terms_lower	additive_terms_upper	yearly	yearly_lower	yearly_upper	multiplicative_terms
65	2021-04-25	1.465682e+06	1.203754e+06	1.367811e+06	1.465489e+06	1.465820e+06	-177396.077126	-177396.077126	-177396.077126	-177396.077126	-177396.077126	-177396.077126	0.0
66	2021-07-25	1.514529e+06	1.458420e+06	1.625763e+06	1.513895e+06	1.515037e+06	28816.695636	28816.695636	28816.695636	28816.695636	28816.695636	28816.695636	0.0
67	2021-09-09	1.539220e+06	1.474853e+06	1.652073e+06	1.537885e+06	1.540367e+06	25957.575229	25957.575229	25957.575229	25957.575229	25957.575229	25957.575229	0.0
68	2021-10-10	1.555860e+06	1.447880e+06	1.625942e+06	1.553616e+06	1.557870e+06	-21684.229386	-21684.229386	-21684.229386	-21684.229386	-21684.229386	-21684.229386	0.0
69	2021-11-13	1.574111e+06	1.455854e+06	1.630780e+06	1.570925e+06	1.576976e+06	-32109.060893	-32109.060893	-32109.060893	-32109.060893	-32109.060893	-32109.060893	0.0

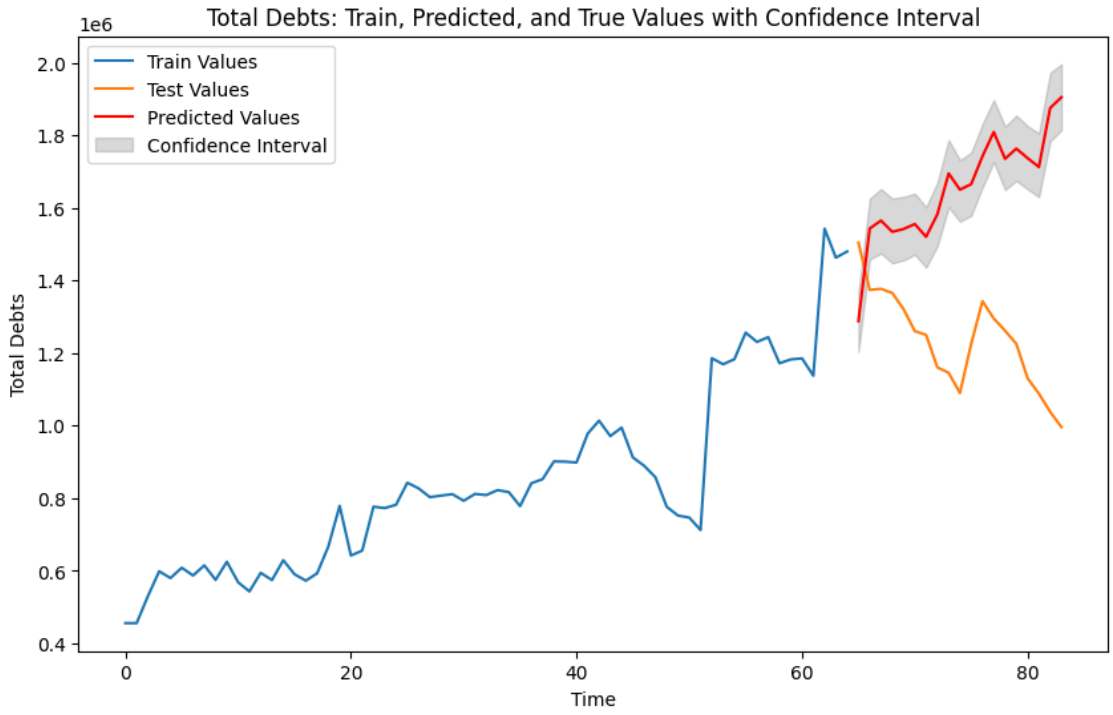
```
69  -32109.060893  -32109.060893  -32109.060893                  0.0

    multiplicative_terms_lower  multiplicative_terms_upper   Total_debts
65                         0.0                         0.0  1.288286e+06
66                         0.0                         0.0  1.543345e+06
67                         0.0                         0.0  1.565178e+06
68                         0.0                         0.0  1.534176e+06
69                         0.0                         0.0  1.542002e+06
```

[12]:
```
rmse = np.sqrt(np.mean((prophet_prediction_df[['Total_debts']].values -␣
 ↪test_df[['Total_debts']].values) ** 2))
print("RMSE is: ", rmse)
```

```
RMSE is:  489755.14000579435
```

[13]:
```
train_values = train_df['Total_debts']
test_values = test_df['Total_debts']
# Creating a plot that includes the predicted values, true values, and␣
 ↪confidence intervals.
plot_predictions(train_values, test_values, forecast["yhat"],␣
 ↪forecast["yhat_lower"], forecast["yhat_upper"])
```



1.4 3. Executing baseline machine learning models

In this part, we use machine learning Linear Regression, Forest Regressor, Boosting Regressor, and Decision Tree Regressor to find out the most effective models. By analyzing historical total debt data and employing advanced machine learning techniques, we aim to offer valuable insights into predicting future total debts.

- **The steps are implemented:**

1. **Split data use time-series cross-validation**: In time series cross-validation, the dataset is split into multiple folds based on time. I use the rolling window approach, a fixed-size test window is moved forward in time
2. **Run Machine Learning Models**: In this step, we train model machine learning in each fold and write the results for evaluation in next the step.
3. **Model evaluation**: We use metrics to evaluate such as root mean squared error (RMSE), mean absolute error (MAE), mean squared error (MSE), and R2 score (R2_score).

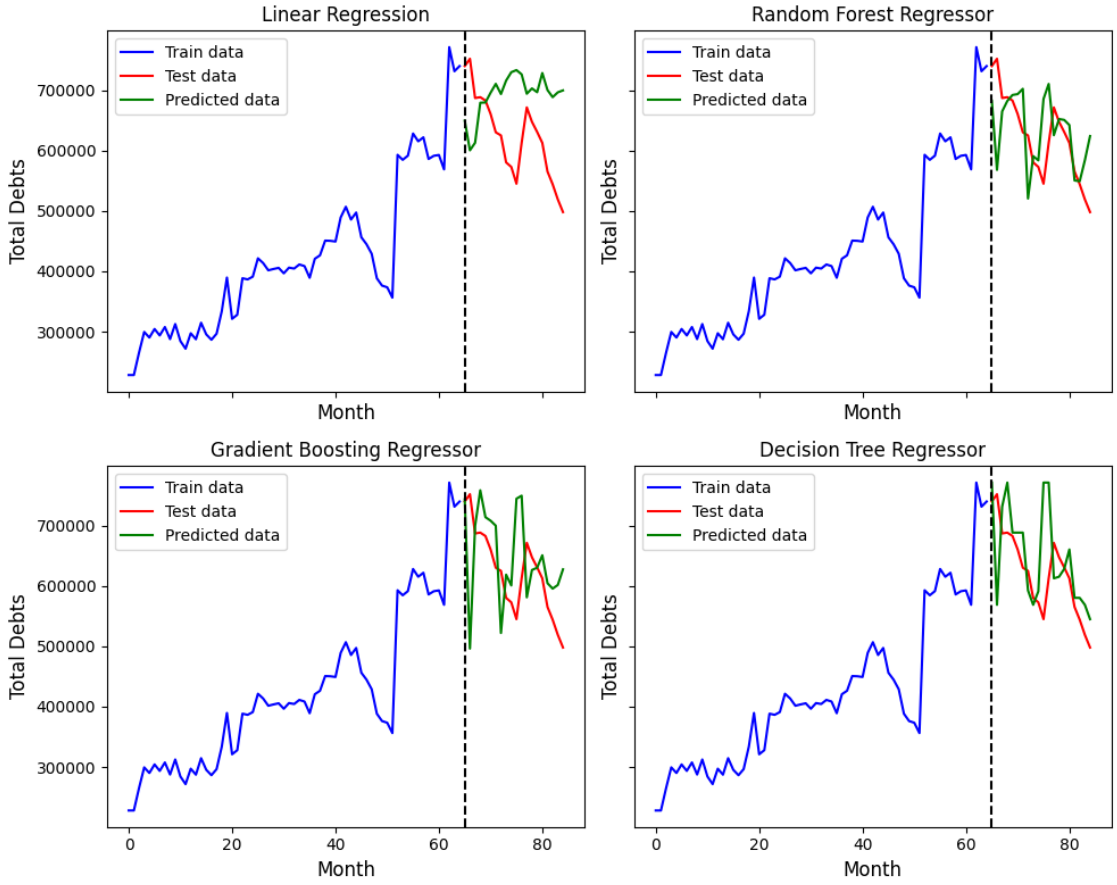
[14]:
```
dataset_generation = MachineLearningModel()
dataset_generation.run_ml_models_results()
```

```
2023-11-16 09:49:35.265295  File date_smedebtsu.csv
2023-11-16 09:49:36.125340  File lag3_smedebtsu.csv
2023-11-16 09:49:36.946566  File lag3_date_smedebtsu.csv
```

1.4.1 Visualizing the predicted results

Dataset: "date_smedebtsu"

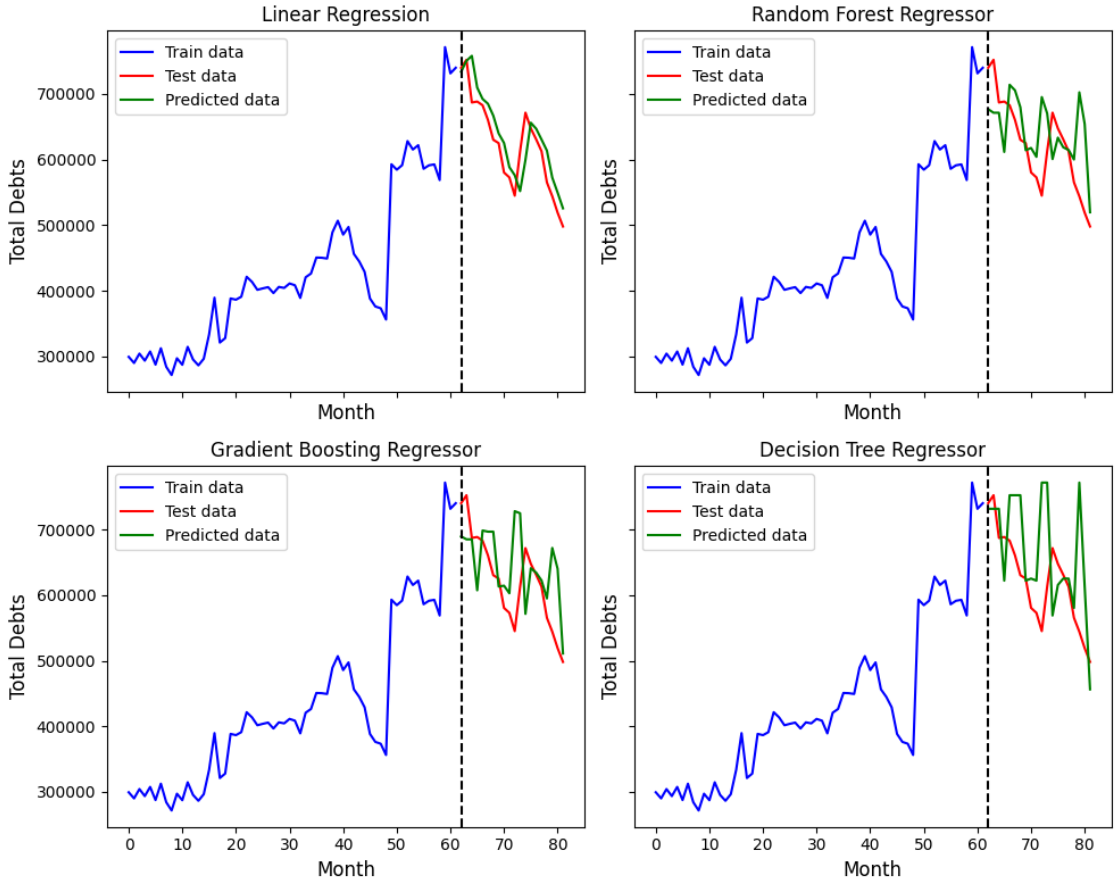
[15]:
```
date_results_path = "../../results/date_smedebtsu/ML_regression_models/
 ↪average_results.json"
dataset_generation.plot_synthesize_charts(date_results_path)
```



<Figure size 1000x600 with 0 Axes>

Dataset: "lag3_smedebtsu"

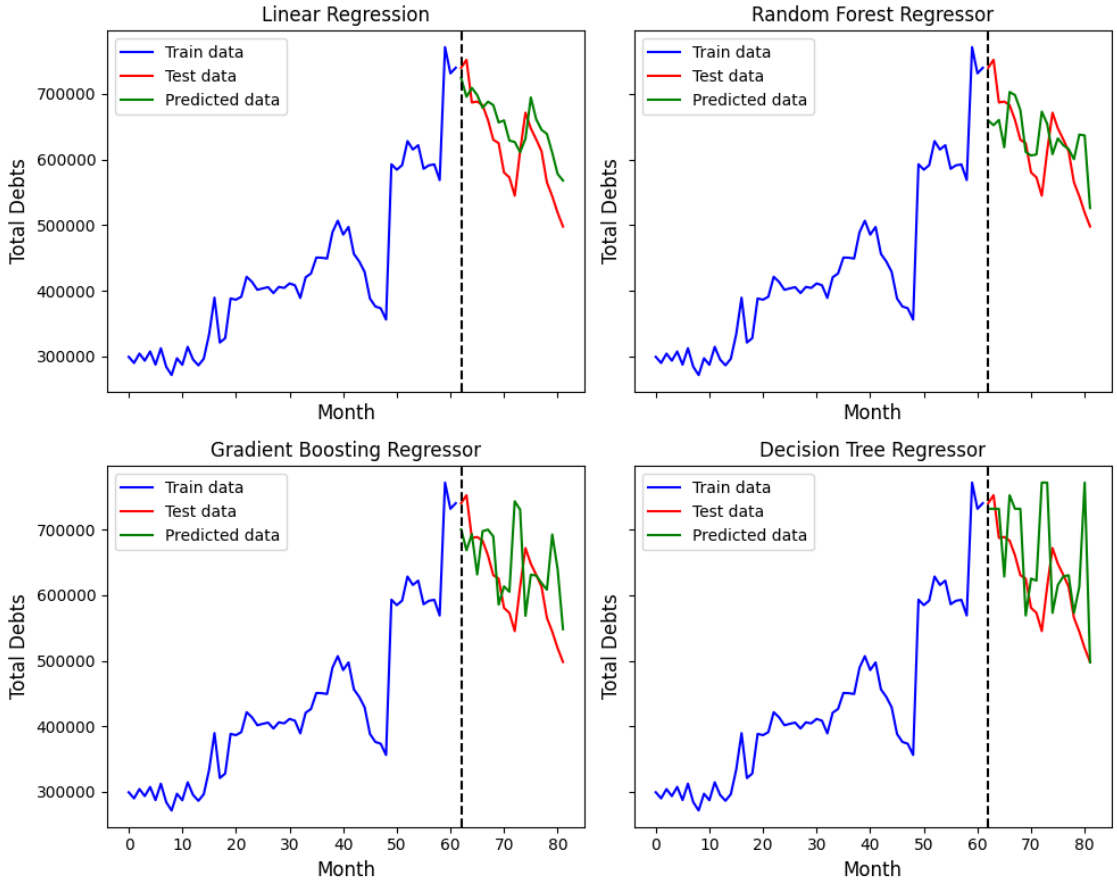
[16]:
```
date_results_path = "../../results/lag3_smedebtsu/ML_regression_models/
 ↪average_results.json"
dataset_generation.plot_synthesize_charts(date_results_path)
```



<Figure size 1000x600 with 0 Axes>

Dataset: "lag3_date_smedebtsu"

[17]:
```
date_results_path = "../../results/lag3_date_smedebtsu/ML_regression_models/
↪average_results.json"
dataset_generation.plot_synthesize_charts(date_results_path)
```



<Figure size 1000x600 with 0 Axes>

1.5 4. Train forecasting models using GRU model

implementing included 4 steps:

1. Split data into train/ validation/ test base on a day '2021-04-25'
2. Building model
3. Evaluation test sets use MAE, MSE, RMSE, R2_score metrics
4. Visualization of the results

[18]:
```
rgu_model()
```

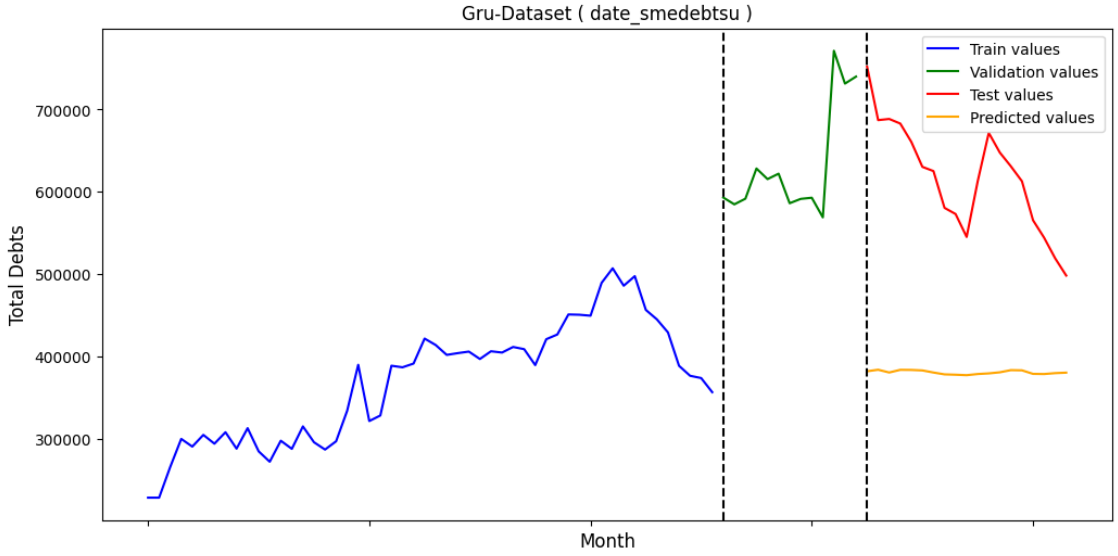
```
2023-11-16 09:49:39.958282  File date_smedebtsu.csv
The number of training samples:  52
The number of validation samples:  13
The number of testing samples:  19
Starting training...
Epoch 1/10
4/4 [==============================] - 3s 161ms/step - loss: 0.8872 -
mean_absolute_error: 0.7745 - val_loss: 0.5579 - val_mean_absolute_error: 0.7362
```

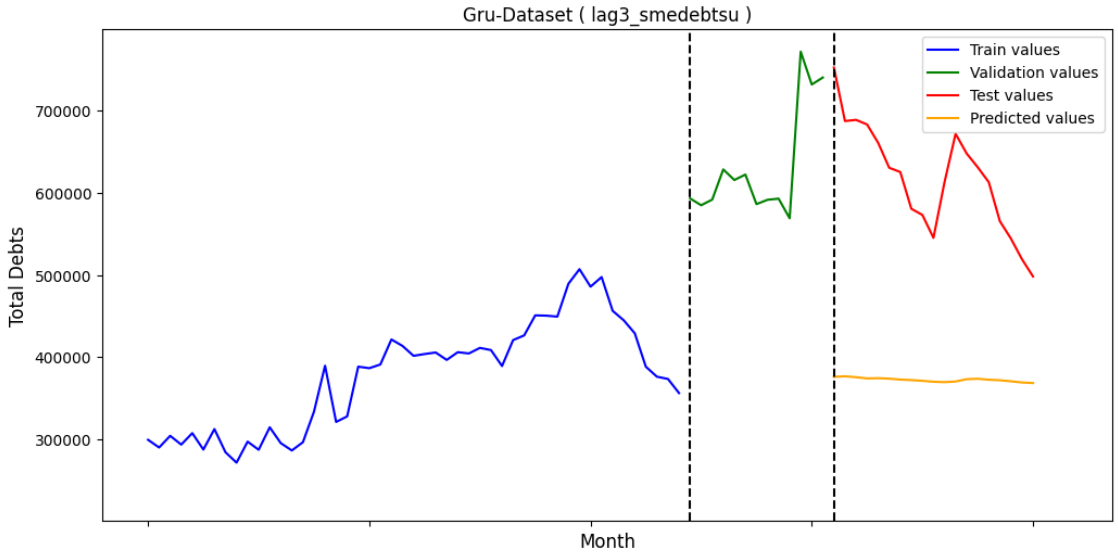
```
Epoch 2/10
4/4 [==============================] - 0s 13ms/step - loss: 0.0867 -
mean_absolute_error: 0.2456 - val_loss: 0.4034 - val_mean_absolute_error: 0.6243
Epoch 3/10
4/4 [==============================] - 0s 13ms/step - loss: 0.1762 -
mean_absolute_error: 0.3535 - val_loss: 0.3379 - val_mean_absolute_error: 0.5696
Epoch 4/10
4/4 [==============================] - 0s 13ms/step - loss: 0.1608 -
mean_absolute_error: 0.3379 - val_loss: 0.3181 - val_mean_absolute_error: 0.5515
Epoch 5/10
4/4 [==============================] - 0s 13ms/step - loss: 0.0366 -
mean_absolute_error: 0.1440 - val_loss: 0.3107 - val_mean_absolute_error: 0.5438
Epoch 6/10
4/4 [==============================] - 0s 13ms/step - loss: 0.0256 -
mean_absolute_error: 0.1320 - val_loss: 0.2963 - val_mean_absolute_error: 0.5298
Epoch 7/10
4/4 [==============================] - 0s 13ms/step - loss: 0.0498 -
mean_absolute_error: 0.1846 - val_loss: 0.2740 - val_mean_absolute_error: 0.5085
Epoch 8/10
4/4 [==============================] - 0s 14ms/step - loss: 0.0284 -
mean_absolute_error: 0.1375 - val_loss: 0.2527 - val_mean_absolute_error: 0.4876
Epoch 9/10
4/4 [==============================] - 0s 13ms/step - loss: 0.0113 -
mean_absolute_error: 0.0861 - val_loss: 0.2363 - val_mean_absolute_error: 0.4710
Epoch 10/10
4/4 [==============================] - 0s 13ms/step - loss: 0.0196 -
mean_absolute_error: 0.1113 - val_loss: 0.2311 - val_mean_absolute_error: 0.4657
Testing model
1/1 [==============================] - 0s 385ms/step
2023-11-16 09:49:44.164908  File lag3_smedebtsu.csv
The number of training samples:  49
The number of validation samples:  13
The number of testing samples:  19
Starting training...
Epoch 1/10
4/4 [==============================] - 3s 153ms/step - loss: 0.1480 -
mean_absolute_error: 0.3034 - val_loss: 0.3785 - val_mean_absolute_error: 0.6024
Epoch 2/10
4/4 [==============================] - 0s 12ms/step - loss: 0.0711 -
mean_absolute_error: 0.2178 - val_loss: 0.2933 - val_mean_absolute_error: 0.5272
Epoch 3/10
4/4 [==============================] - 0s 12ms/step - loss: 0.0417 -
mean_absolute_error: 0.1634 - val_loss: 0.2765 - val_mean_absolute_error: 0.5101
Epoch 4/10
4/4 [==============================] - 0s 12ms/step - loss: 0.0103 -
mean_absolute_error: 0.0839 - val_loss: 0.2697 - val_mean_absolute_error: 0.5025
Epoch 5/10
4/4 [==============================] - 0s 12ms/step - loss: 0.0259 -
```

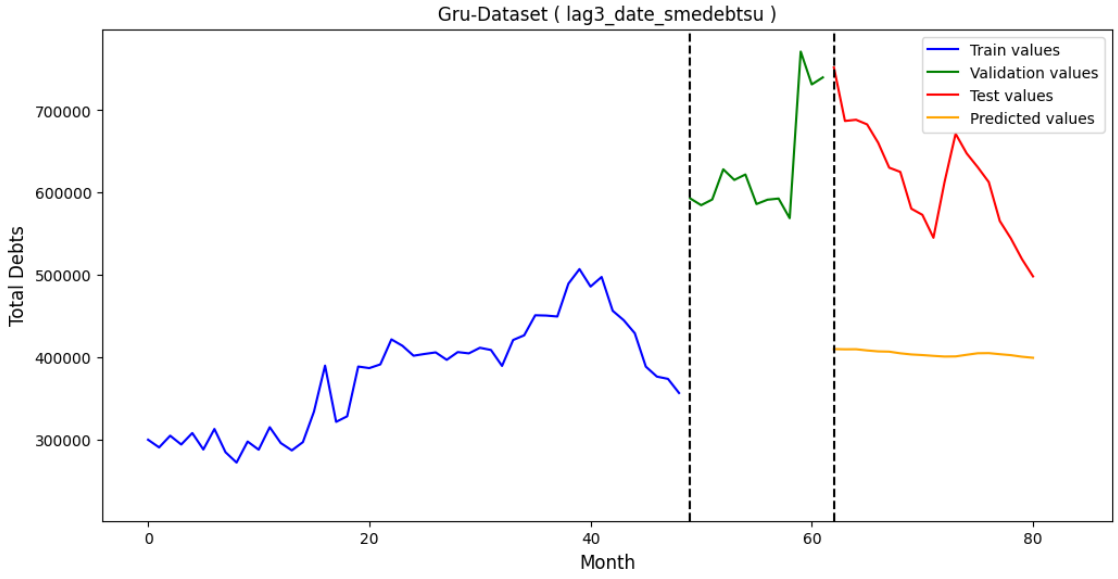
```
mean_absolute_error: 0.1365 - val_loss: 0.2346 - val_mean_absolute_error: 0.4667
Epoch 6/10
4/4 [==============================] - 0s 12ms/step - loss: 0.0094 -
mean_absolute_error: 0.0816 - val_loss: 0.2087 - val_mean_absolute_error: 0.4390
Epoch 7/10
4/4 [==============================] - 0s 13ms/step - loss: 0.0109 -
mean_absolute_error: 0.0849 - val_loss: 0.2112 - val_mean_absolute_error: 0.4420
Epoch 8/10
4/4 [==============================] - 0s 12ms/step - loss: 0.0093 -
mean_absolute_error: 0.0763 - val_loss: 0.2364 - val_mean_absolute_error: 0.4692
Epoch 9/10
4/4 [==============================] - 0s 13ms/step - loss: 0.0061 -
mean_absolute_error: 0.0622 - val_loss: 0.2699 - val_mean_absolute_error: 0.5032
Epoch 10/10
4/4 [==============================] - 0s 13ms/step - loss: 0.0069 -
mean_absolute_error: 0.0722 - val_loss: 0.2895 - val_mean_absolute_error: 0.5226
Testing model
1/1 [==============================] - 0s 367ms/step
2023-11-16 09:49:48.035966  File lag3_date_smedebtsu.csv
The number of training samples:  49
The number of validation samples:  13
The number of testing samples:  19
Starting training...
Epoch 1/10
4/4 [==============================] - 3s 152ms/step - loss: 0.1089 -
mean_absolute_error: 0.2577 - val_loss: 0.4786 - val_mean_absolute_error: 0.6787
Epoch 2/10
4/4 [==============================] - 0s 13ms/step - loss: 0.1170 -
mean_absolute_error: 0.2881 - val_loss: 0.3740 - val_mean_absolute_error: 0.5972
Epoch 3/10
4/4 [==============================] - 0s 13ms/step - loss: 0.0180 -
mean_absolute_error: 0.1141 - val_loss: 0.2596 - val_mean_absolute_error: 0.4938
Epoch 4/10
4/4 [==============================] - 0s 13ms/step - loss: 0.0255 -
mean_absolute_error: 0.1268 - val_loss: 0.2090 - val_mean_absolute_error: 0.4398
Epoch 5/10
4/4 [==============================] - 0s 13ms/step - loss: 0.0263 -
mean_absolute_error: 0.1290 - val_loss: 0.2064 - val_mean_absolute_error: 0.4359
Epoch 6/10
4/4 [==============================] - 0s 13ms/step - loss: 0.0065 -
mean_absolute_error: 0.0650 - val_loss: 0.2075 - val_mean_absolute_error: 0.4365
Epoch 7/10
4/4 [==============================] - 0s 13ms/step - loss: 0.0163 -
mean_absolute_error: 0.1037 - val_loss: 0.1982 - val_mean_absolute_error: 0.4259
Epoch 8/10
4/4 [==============================] - 0s 14ms/step - loss: 0.0092 -
mean_absolute_error: 0.0778 - val_loss: 0.1900 - val_mean_absolute_error: 0.4168
Epoch 9/10
```

```
4/4 [==============================] - 0s 14ms/step - loss: 0.0092 -
mean_absolute_error: 0.0785 - val_loss: 0.2023 - val_mean_absolute_error: 0.4315
Epoch 10/10
4/4 [==============================] - 0s 13ms/step - loss: 0.0068 -
mean_absolute_error: 0.0609 - val_loss: 0.2304 - val_mean_absolute_error: 0.4627
Testing model
1/1 [==============================] - 0s 365ms/step
```

[19]:
```
plot_gru_synthesize_charts()
```


Gru-Dataset (date_smedebtsu)
Train values
Validation values
Test values
Predicted values
Total Debts
Month


Gru-Dataset (lag3_smedebtsu)
Train values
Validation values
Test values
Predicted values
Total Debts
Month


Gru-Dataset (lag3_date_smedebtsu)
Train values
Validation values
Test values
Predicted values
Total Debts
Month

```
<Figure size 640x480 with 0 Axes>
```

1.6 5. Experimental results

- We will conduct an analysis using various base machine learning and deep learning models on three datasets: 'date_smedebtsu,' 'lag3_smedebtsu,' and 'lag3_date_smedebtsu'

1.6.1 MSE

[20]:
```
# Get MSE metrics
mse_benchmark = get_metric_files(result_path, method="MSE")
mse_benchmark_df = pd.DataFrame(mse_benchmark)
mse_benchmark_df
```

[20]:

	linear_regression	random_forest_regressor	gradient_boosting_regressor	\
0	1.365920e+10	5.417036e+09	9.401521e+09	
1	2.439622e+09	3.759277e+09	6.278216e+09	
2	1.271705e+09	5.047607e+09	5.494470e+09	

	decision_tree_regressor	RGU	dataset
0	6.930855e+09	6.009354e+10	date_smedebtsu
1	9.485647e+09	4.921908e+10	lag3_date_smedebtsu
2	9.497935e+09	6.385070e+10	lag3_smedebtsu

1.6.2 MAE

[21]:
```
# Get MAE metrics
mae_benchmark = get_metric_files(result_path, method="MAE")
mae_benchmark_df = pd.DataFrame(mae_benchmark)
mae_benchmark_df
```

[21]:

	linear_regression	random_forest_regressor	gradient_boosting_regressor	\
0	101287.419727	52977.376540	72503.823437	
1	42926.306977	49500.610500	60359.834664	
2	29404.894820	54939.694755	55014.010137	

	decision_tree_regressor	RGU	dataset
0	58542.9515	236834.038947	date_smedebtsu
1	69535.3560	213020.986316	lag3_date_smedebtsu
2	71789.0470	244807.395855	lag3_smedebtsu

1.6.3 RMSE

[22]:
```
result_path = "../../results/"
rmse_benchmark = get_metric_files(result_path, method="RMSE")
# Get RMSE metrics
rmse_benchmark_df = pd.DataFrame(rmse_benchmark)
rmse_benchmark_df
```

[22]:

	linear_regression	random_forest_regressor	gradient_boosting_regressor	decision_tree_regressor	RGU	dataset
0	108515.322793	69875.588253	91625.667664	71955.728232	245139.837065	date_smedebtsu
1	46937.991933	57679.338221	73310.080863	88182.663058	221853.727526	lag3_date_smedebtsu
2	35139.942282	65745.563870	69373.393981	89318.981727	252686.958912	lag3_smedebtsu

1.6.4 R2 score

[23]:
```
# Get R2_score metrics
R2_score_benchmark = get_metric_files(result_path, method="R2_score")
R2_score_benchmark_df = pd.DataFrame(R2_score_benchmark)
R2_score_benchmark_df
```

[23]:

	linear_regression	random_forest_regressor	gradient_boosting_regressor	decision_tree_regressor	RGU	dataset
0	-21.003883	-7.225288	-13.226448	-9.800898	-13.435846	date_smedebtsu
1	-3.068447	-4.740244	-9.372902	-14.902535	-10.823551	lag3_date_smedebtsu
2	-1.118985	-6.964072	-8.030338	-14.989457	-14.338402	lag3_smedebtsu

[ ]:
```

```