dData

FINAL INTERN PRESENTATION

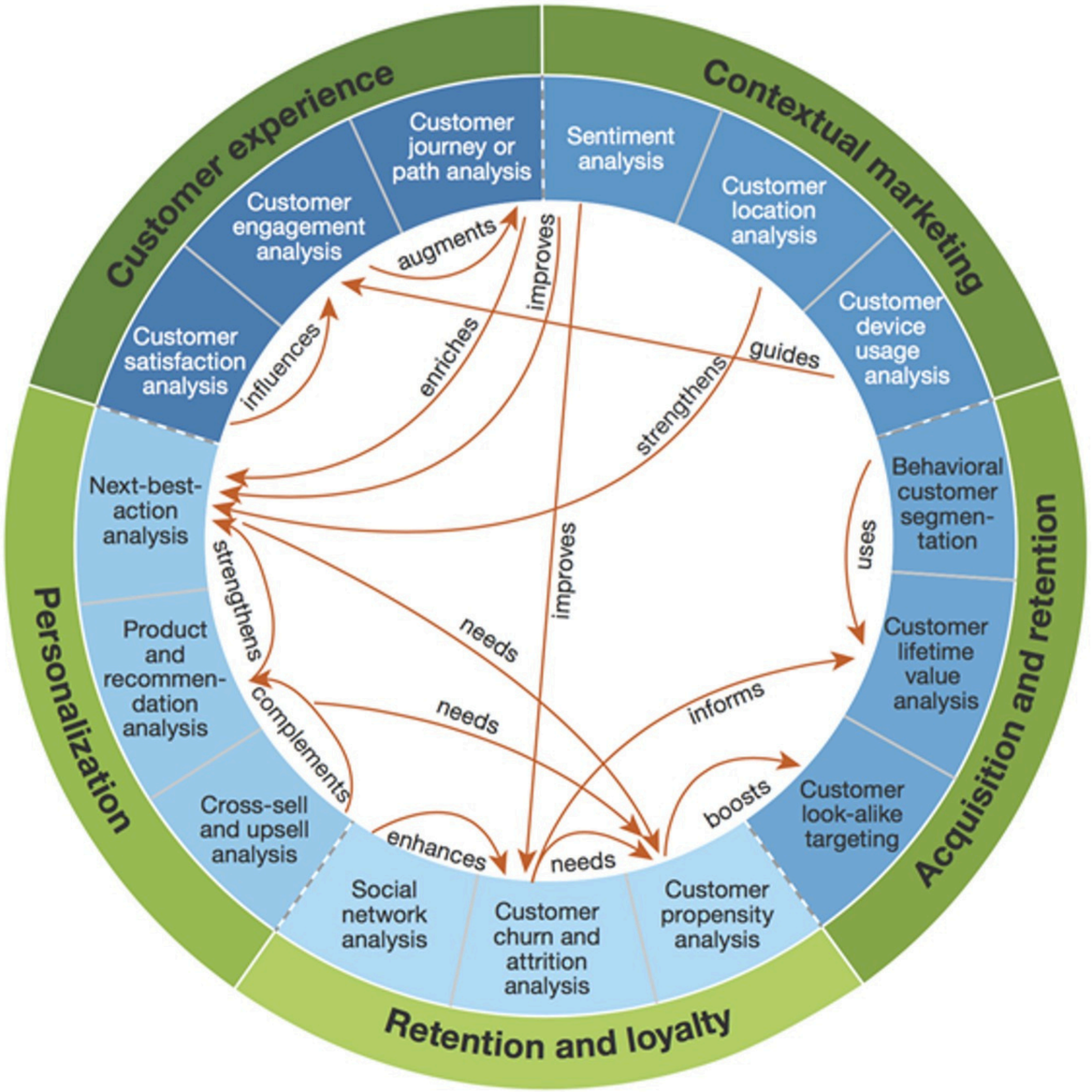
Presenters:

1. Norapath Arjanurak : Third-year Robotics and AI student: Chulalongkorn University
2. Saruch Santhidej : Third-year Robotics and AI student: Chulalongkorn University

TABLE OF CONTENT

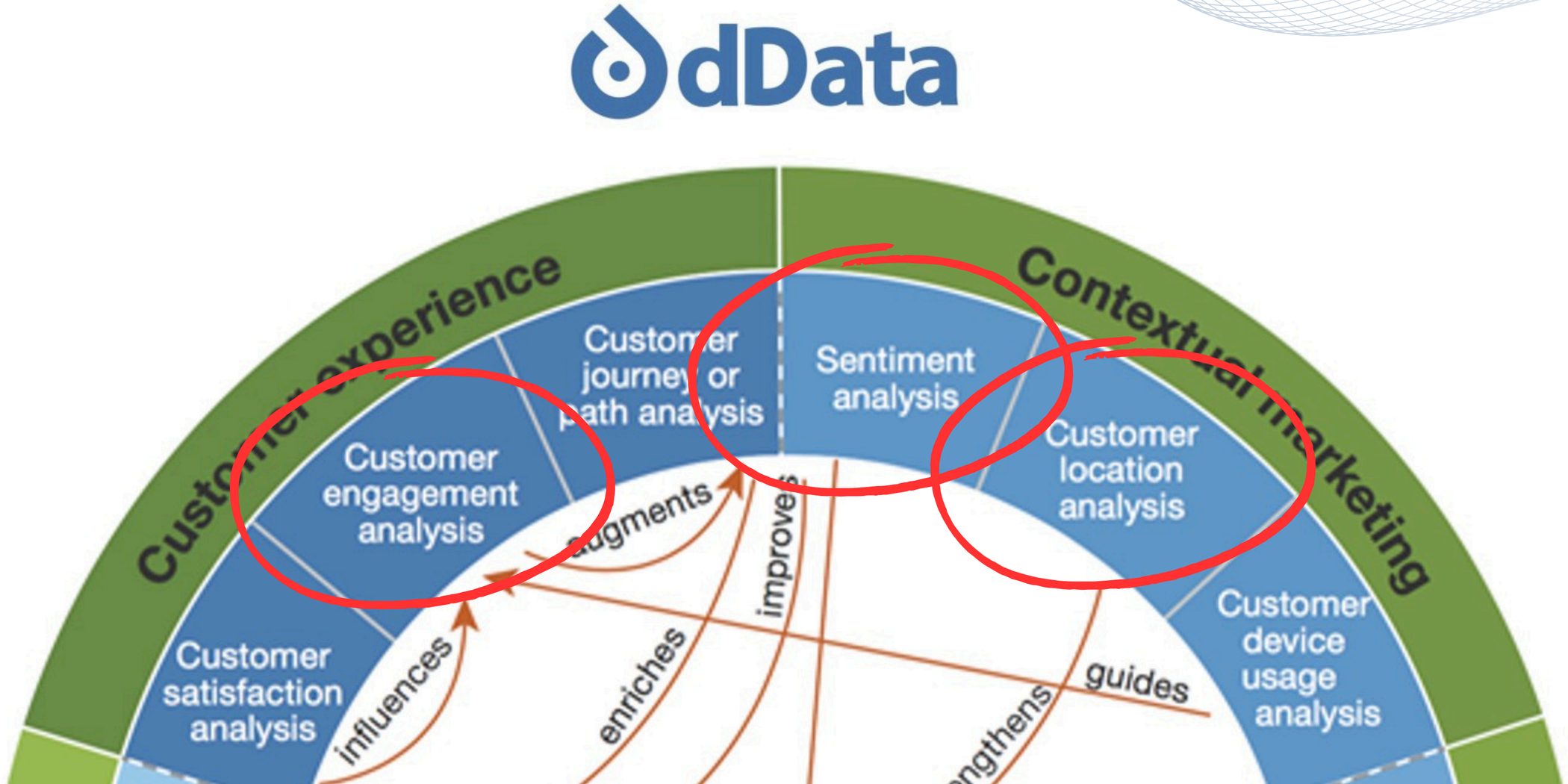
OVERVIEW

dData

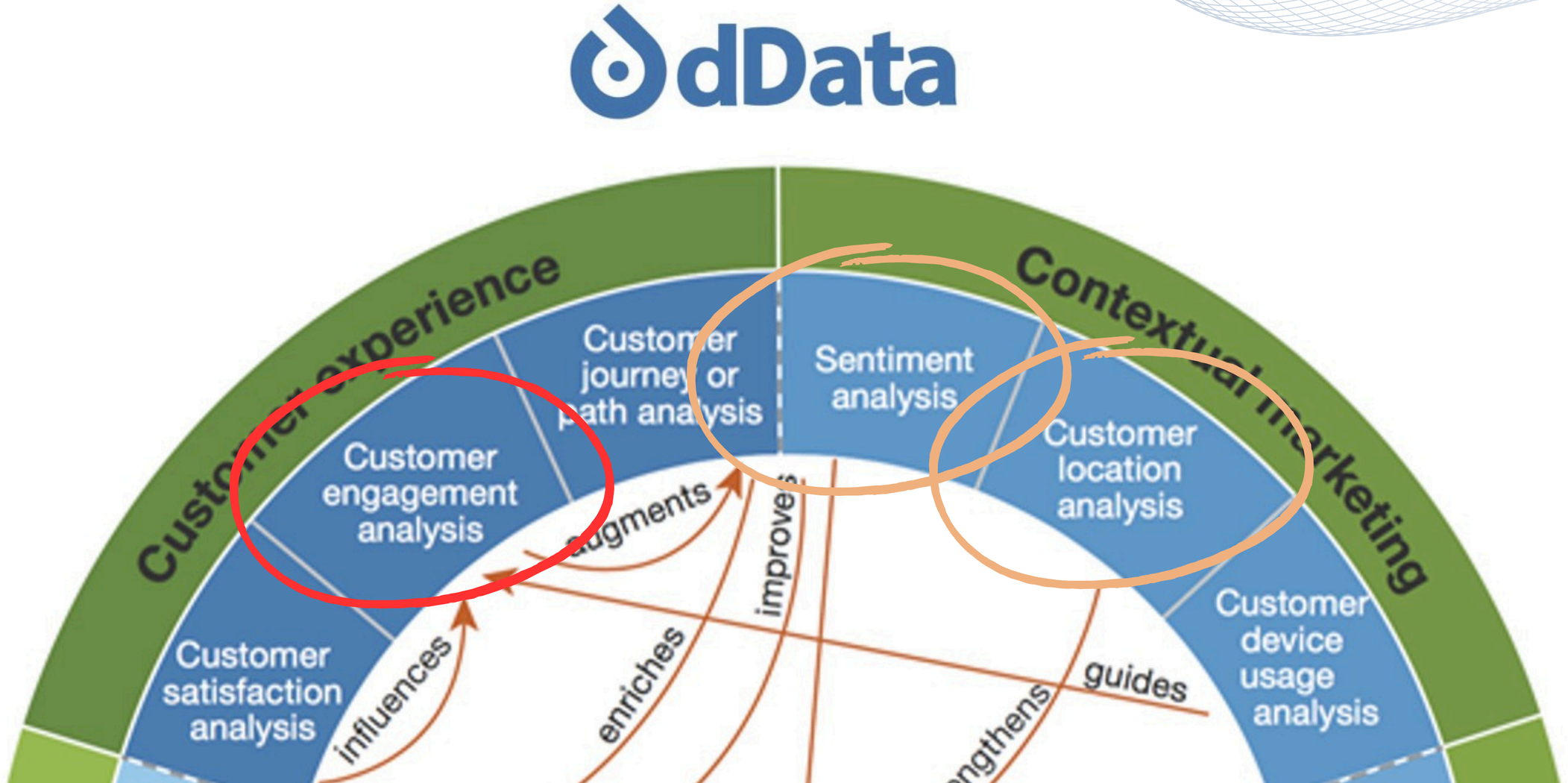


Forrester :15 key customer analytics methods that customer insights (CI) professionals must master.

SELECTION:

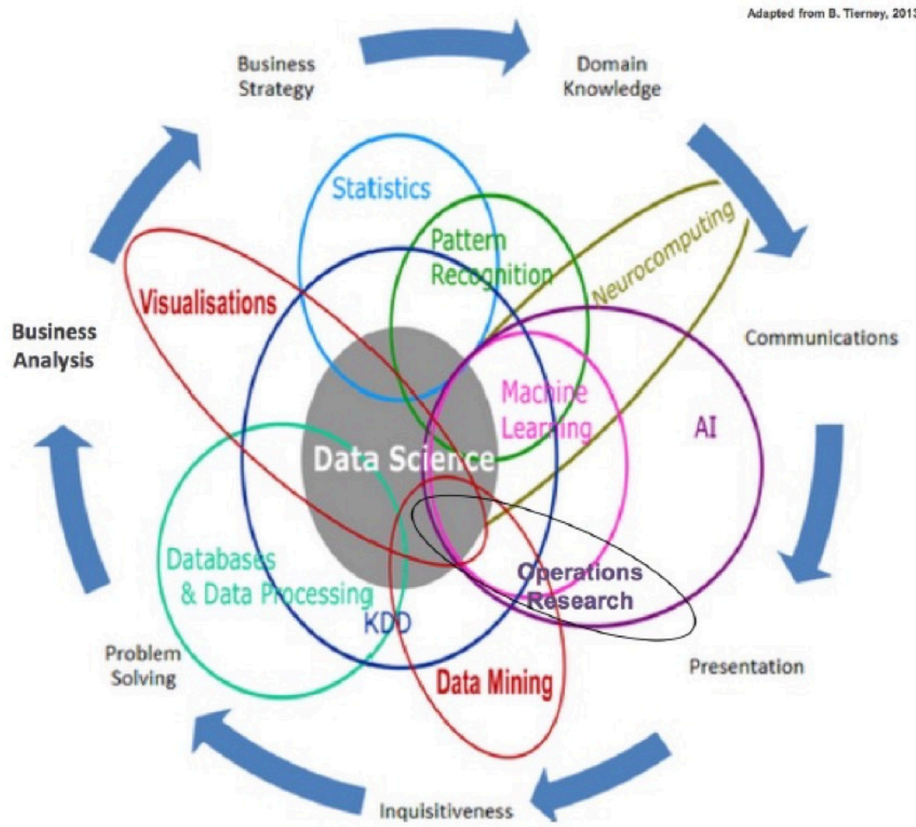

dData
Customer experience
Contextual marketing
Customer engagement analysis
Customer journey or path analysis
Sentiment analysis
Customer location analysis
Customer satisfaction analysis
Customer device usage analysis
augments
improves
influences
enriches
guides
strengthens

SELECTION:


dData
Customer experience
Contextual marketing
Customer engagement analysis
Customer journey or path analysis
Sentiment analysis
Customer location analysis
Customer satisfaction analysis
Customer device usage analysis
augments
improves
influences
enriches
guides
strengthens

CUSTOMER ENGAGEMENT ANALYSIS

Data Science in Multidisciplinary



MEANING

Interaction between a company and its customers

"customers who are more engaged with your brand are more likely to buy or promote your product or service"

https://www.entrepreneurshipinabox.com/24974/how-customer-engagement-analysis-helps-your-marketing-campaign/

ASSUMTION

- We are a **Consult Company named ddData Company**
- Our Client: Food Delivery Start-up who wants to consult regarding investing in a payment system(which brand?Type?Online or Retail) based on customer engagement analysis

WHAT ARE WE DOING?

- Recommend the payment system brand
- Ranking the priority for payment platforms

HOW?

- Recommend the payment system brand--> Compare the past transactions to select the most used one.
- Ranking the priority for payment platforms--> Use the Merchant State data to rank the platforms.

DATA

dData

Card Dataset

- Card ID
- Holder(Users)/Index
- Types: Credit, Debit, Debit(Prepaid)
- Brand: Mastercard, Visa, Amex,Discover
- OpenDate, Expire,Has Chip, On Dark Web

User Dataset

- User_id
- Current/Retirement Ages
- Birth Month/Year
- Yearly Income and Total Debt
- Location(Address/State/ZipCode)

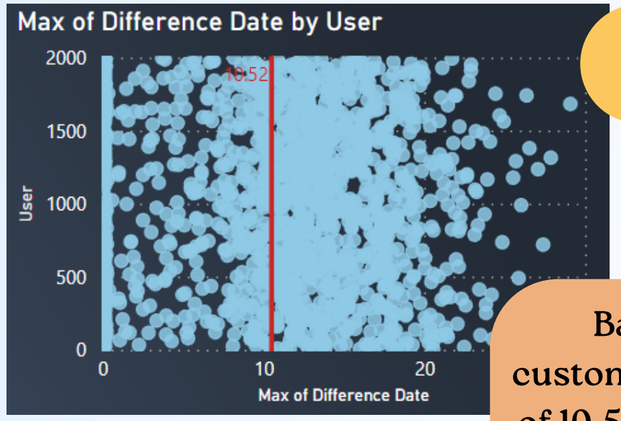
Transaction Dataset

- User_id: Card Index
- Date/Time of Transaction
- MerchantDetail(Name,City:Mostly Online, State)
- Amount of Transaction

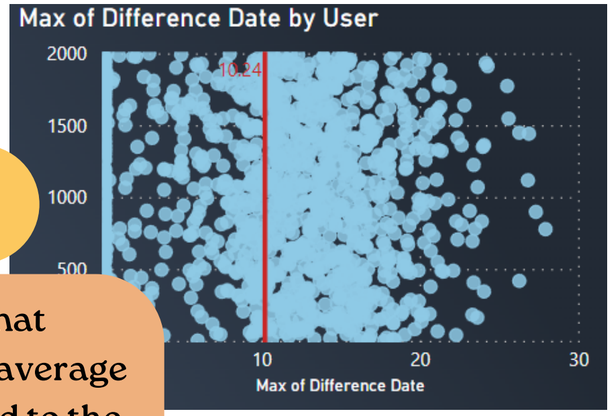
EDA!

SUMMARIZE FROM EDA#1

dData

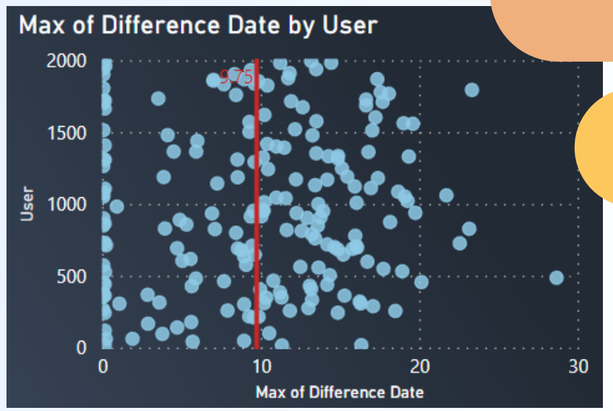


Mastercard

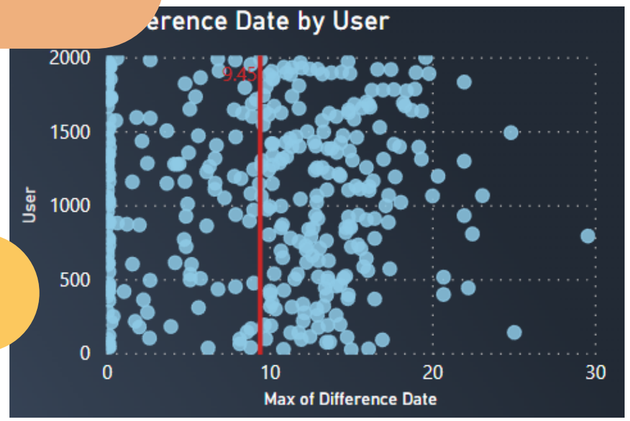


VISA

Based on these visualizations, we can assume that customers hold Mastercard the longest, which is an average of 10.52 year. Indeed, it is not that different compared to the other three brands: VISA, DISCOVER, and AMEX accounting to 10.24, 9.75,and 9.45months, respectively.



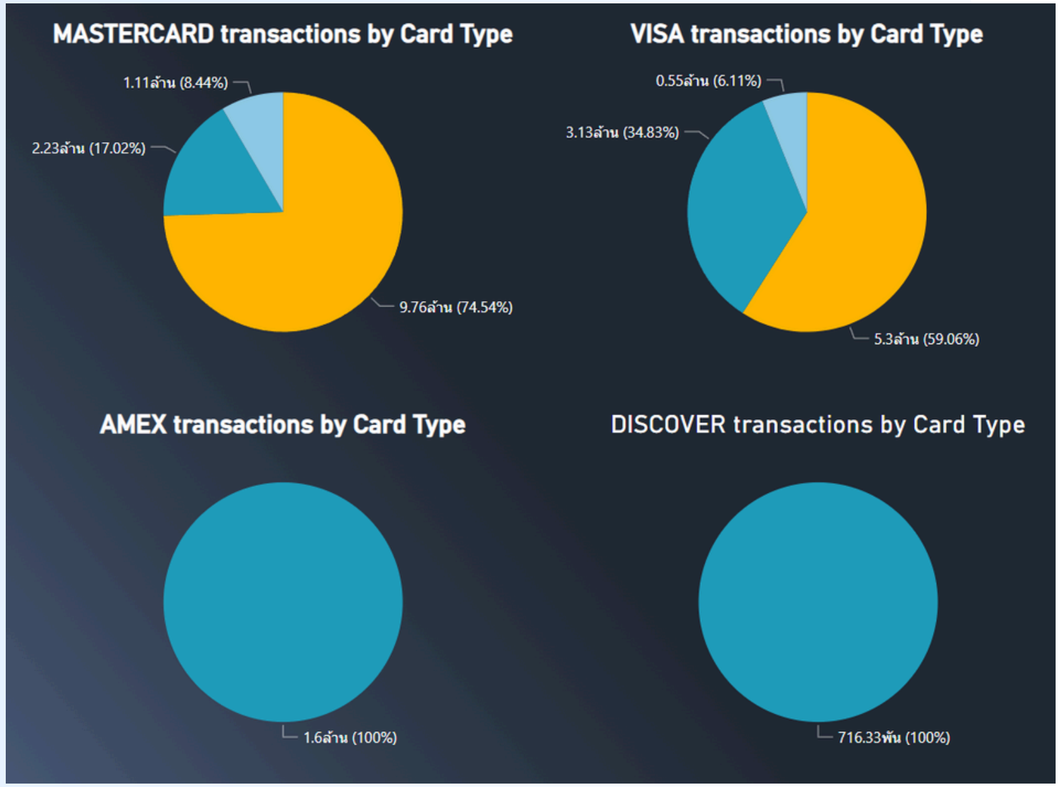
DISCOVER

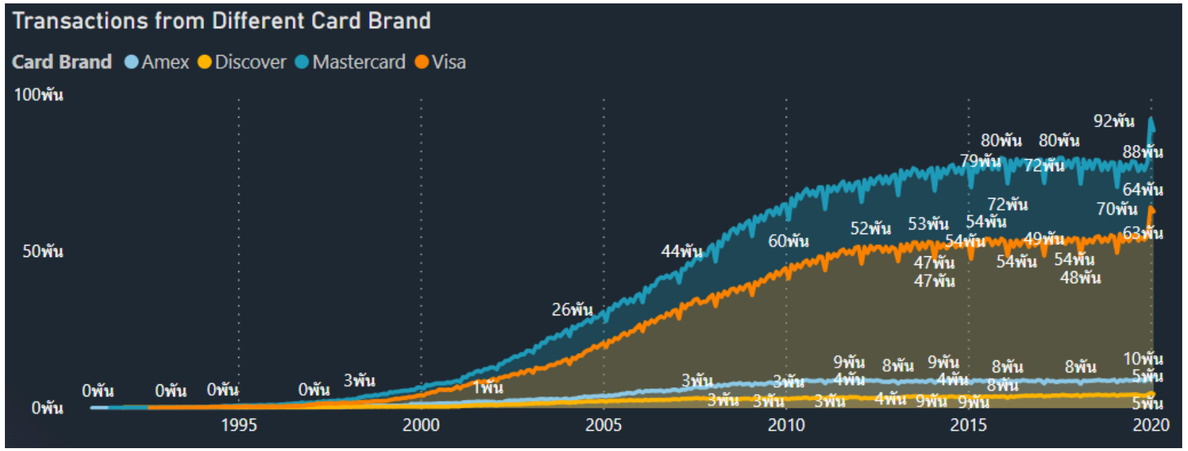


AMEX

INTRODUNCTION | EDA | IMPLEMENT | RESULT | LESSON LEARN

SUMMARIZE FROM EDA#2





The most popular one is the Debit Mastercard which is around 66K transactions at the end of 2020. The second most popular one is the Debit Visa which is lower than Debit Mastercard around 29K transactions(which is A LOT!!).

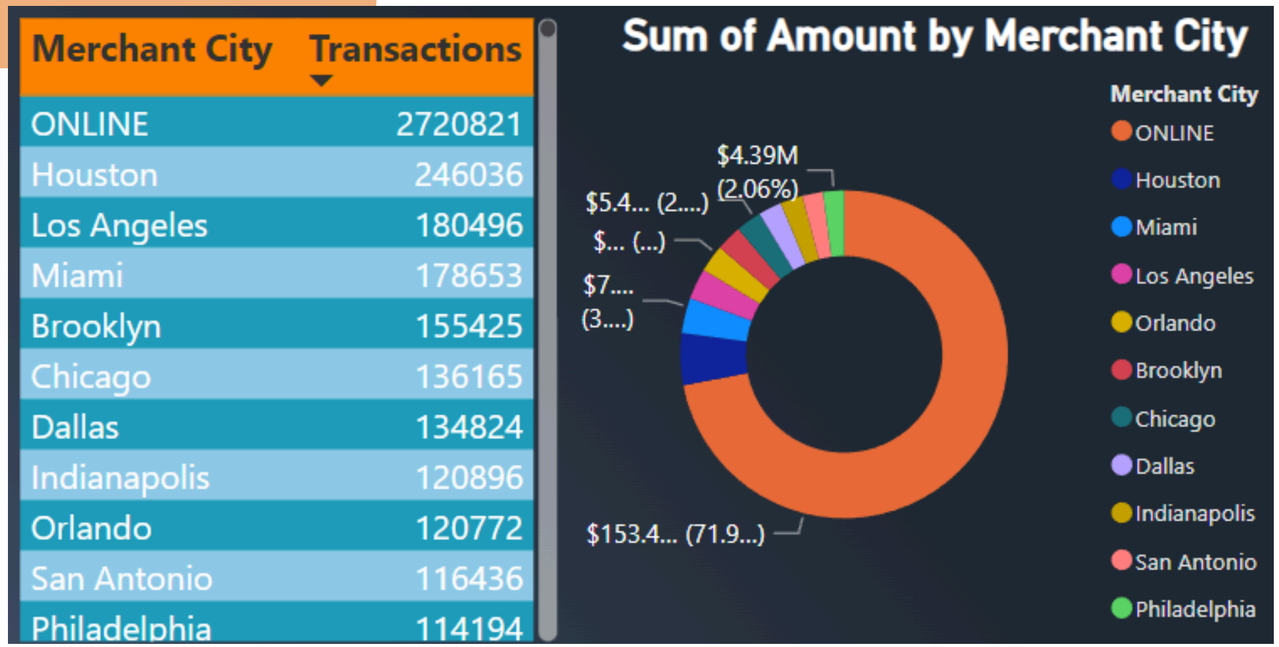
However, for Credit, Visa has the most usage, accounting for 22k at the same time which is higher than Mastercard around 15k.

SUMMARIZE FROM EDA#3

dData

When looking at the total transactions from various purchase state, we can guarantee that most customers prefers to use a Payment card for online shopping which show vast differences from other fields numbered around 2.5M differences

Merchant City	Transactions
ONLINE	2720821
Houston	246036
Los Angeles	180496
Miami	178653
Brooklyn	155425
Chicago	136165
Dallas	134824
Indianapolis	120896
Orlando	120772
San Antonio	116436
Philadelphia	114194



SUMMARIZE

dData

Summarize 1

Mastercard has the most used time.

Summarize 2

Mastercard is the most popular in Debit and Debit(Prepaid). Whereas Visa is the most popular in Credit.

Summarize 3

Customers tend to use the payment card in online transactions.

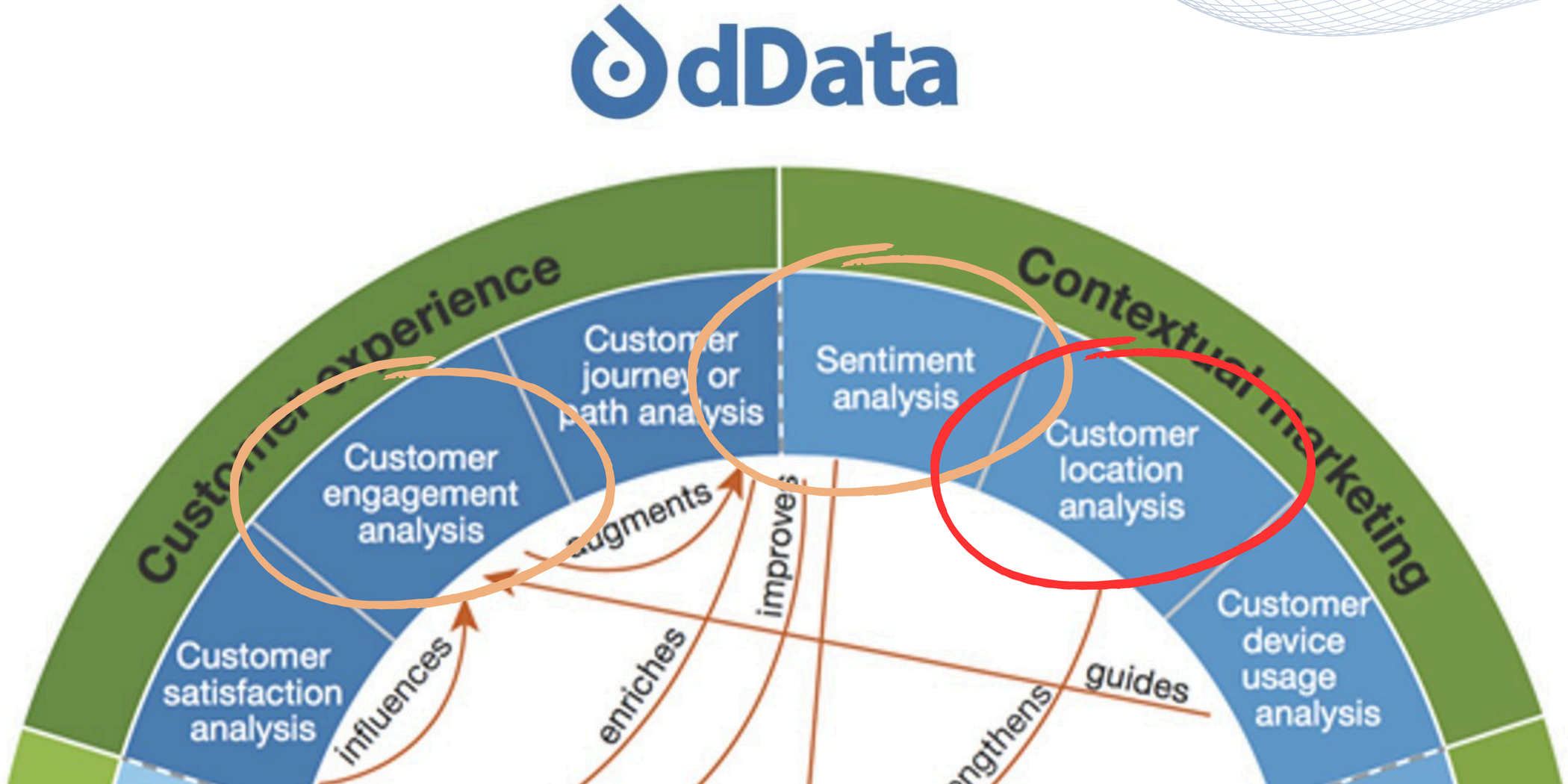
Mastercard is widely used for Debit and Prepaid cards, while Visa is more commonly associated with Credit cards, both being popular choices for online transactions, with customer preferences varying based on factors like convenience and rewards, and regional variations also playing a role.

LESSON LEARN

dData

1. Visualization(Power bi)
2. Handle with Big Data
3. Data Interpretation

SELECTION:


dData
Customer experience
Contextual marketing
Customer engagement analysis
Customer journey or path analysis
Sentiment analysis
Customer location analysis
Customer satisfaction analysis
Customer device usage analysis
augments
improves
influences
enriches
guides
strengthens

Customer Location Analysis

Meaning

Analyse Customer location to gain business insight

"can be used to uniform multiple business decisions, from selecting the next site location of a store or where to place billboard advertisements to reach wider audiences."

https://www.entrepreneurshipinabox.com/24974/how-customer-engagement-analysis-helps-your-marketing-campaign/

Assumption

- We are a **Consult Company named ddData Company**
- Our Client: A group of middlemen who want to receive information about the crop in different states; whether what is the most cultivated crop in each state? The client wants to cut off the logistic expense.

What are we doing?

- Extract data from the google earth engine for cropland.
- Find the business insight.
- Compare from different area.

Tools

dData

Google Earth Engine

PYTHON

Import ee

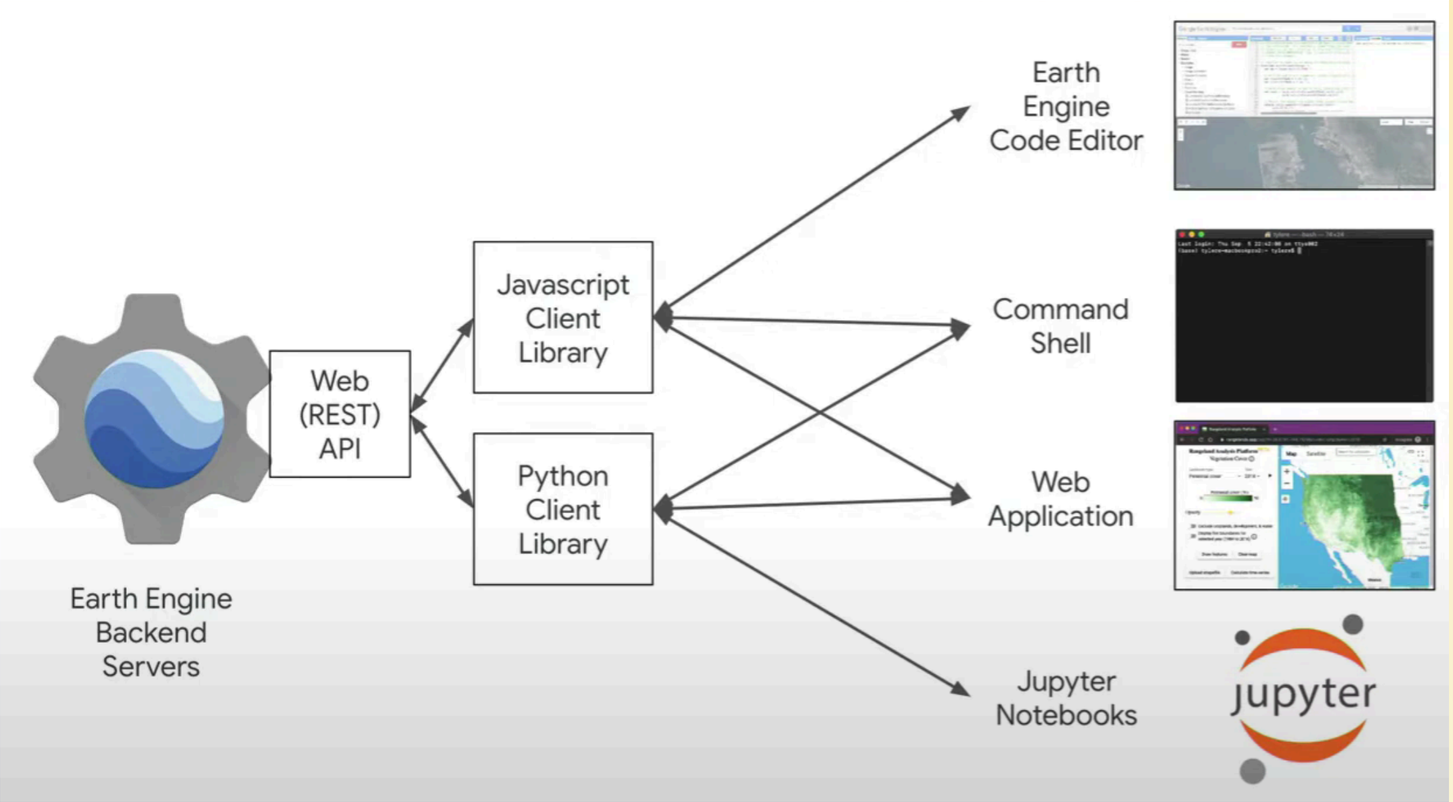
Import google

Import os

ดร. สิทธิศักดิ์ หมูคำหล้า

From Pixels to Insight: Explore the pain point of GGE

Architecture



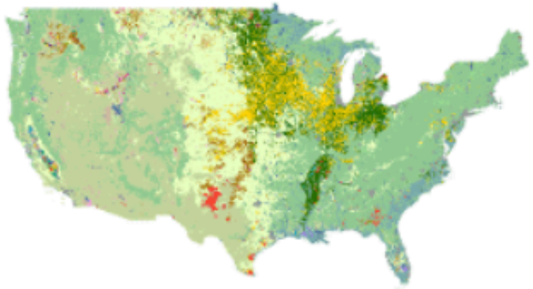
Credit: https://youtu.be/CDyTP5Ao4Bs

Google Earth Engine

dData

Click me!

USDA NASS Cropland Data Layers



Dataset Availability

1997-01-01T00:00:00Z–2022-01-01T00:00:00

Dataset Provider

USDA National Agricultural Statistics Service

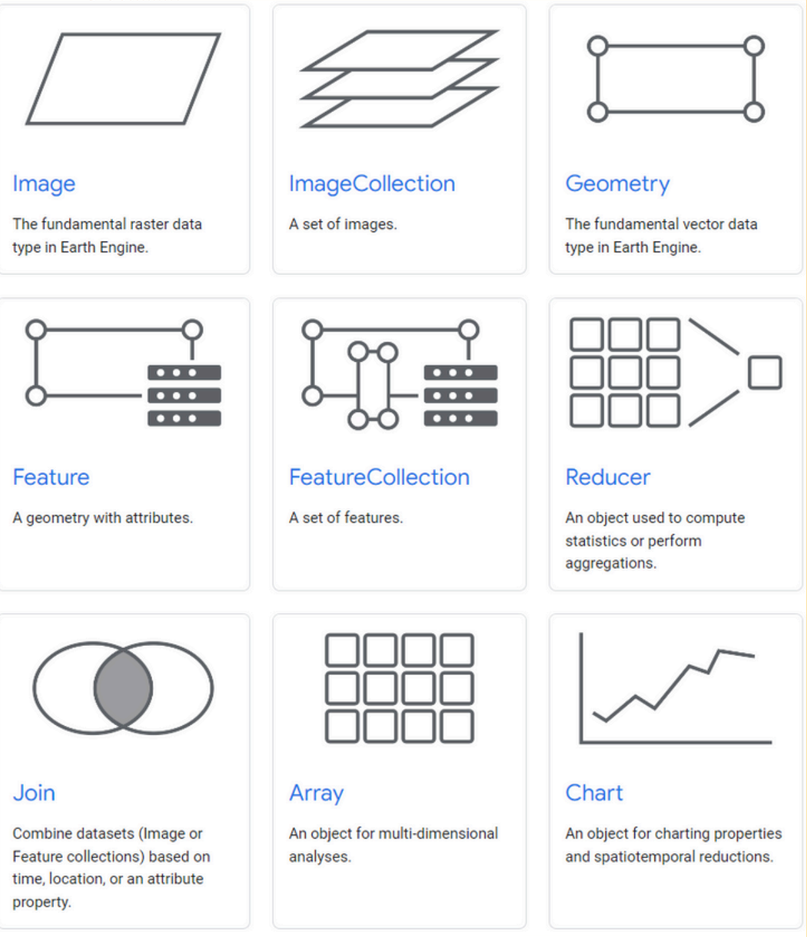
Earth Engine Snippet

```
ee.ImageCollection("USDA/NASS/CDL")
```

Tags

cropland landcover usda cdl nass

Common Earth Engine object classes



Band and Band Table

"Each band has its own name, pixel values, pixel resolution, and projection."

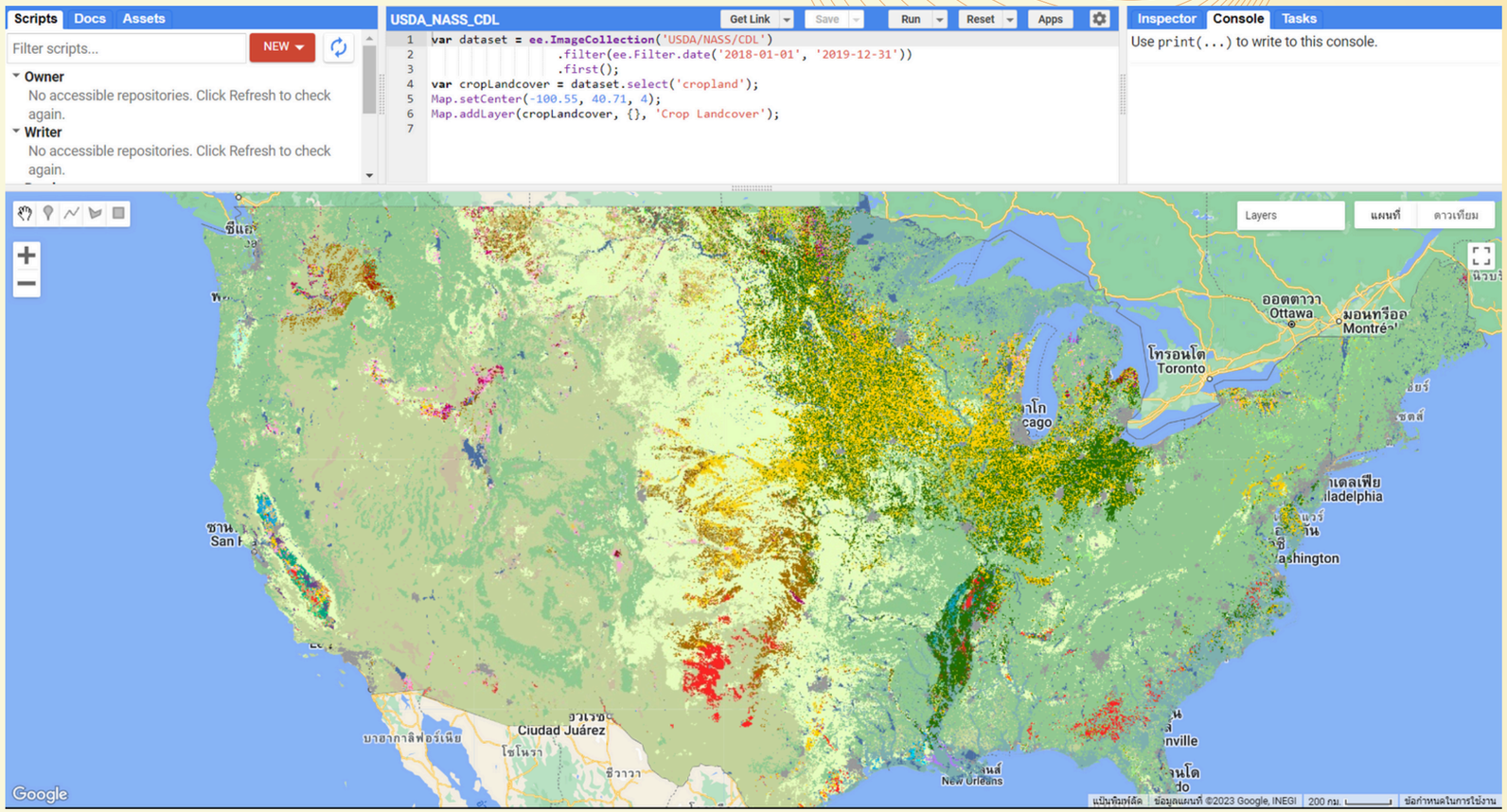
Bands

Name	Min	Max	Description
cropland	1	254	Main crop-specific land cover classification.
cultivated	1	2	Classification layer for identifying cultivated and non-cultivated land cover. Available from 2013 to 2017.
confidence	0	100	Per-pixel predicted confidence of the given classification, with 0 being the least confident and 100 the most confident. Available from 2008 to 2017 (Note: Confidence for Florida and Washington D.C. is unavailable for 2010).

cropland Class Table

Value	Color	Description
1	#ffd300	Corn
2	#ff2626	Cotton
3	#00a8e2	Rice

Earth Engine Code Editor(1)



https://www.nass.usda.gov/Research_and_Science/Cropland/SARS1a.php

Earth Engine Code Editor(2)

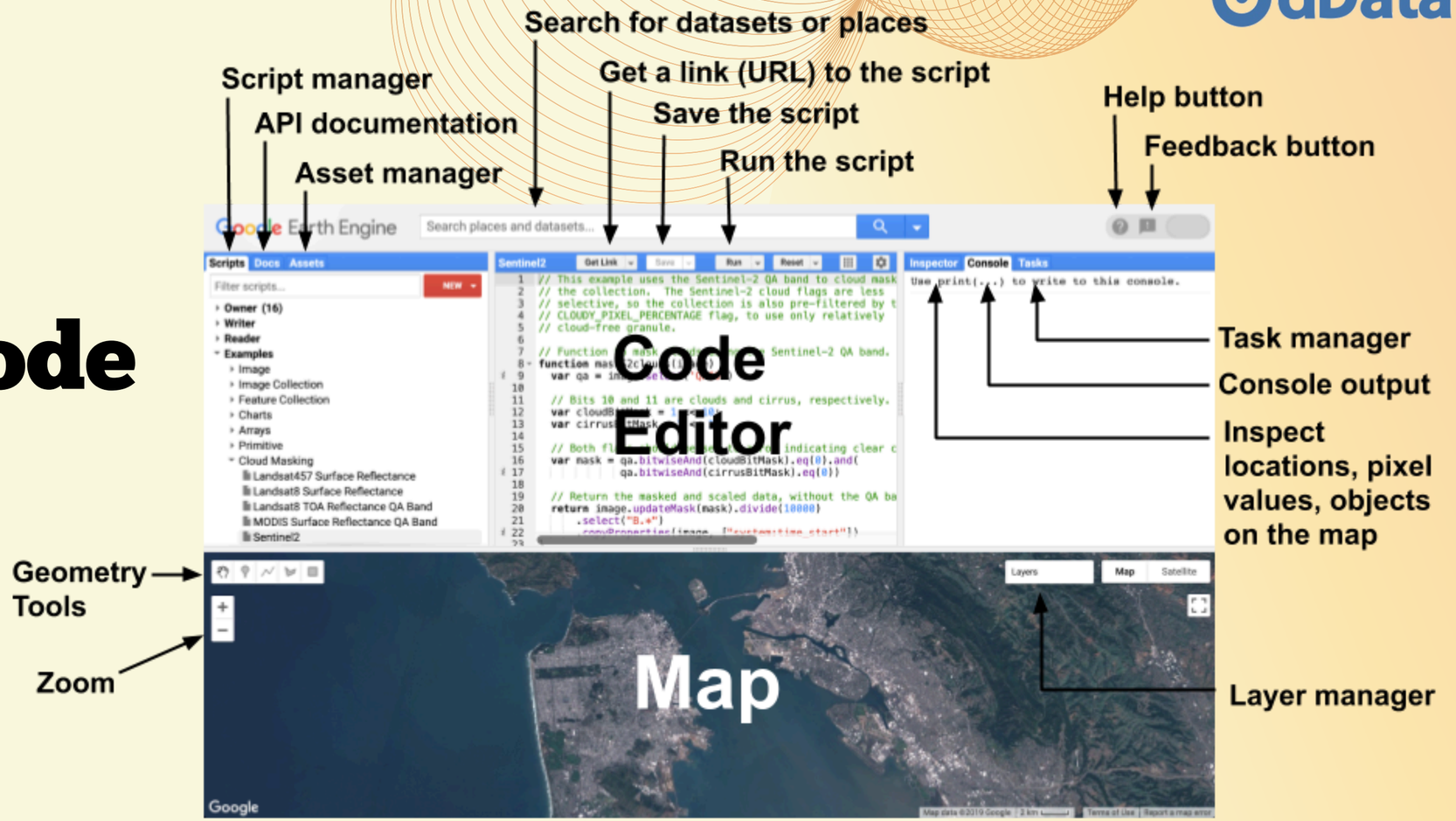


Figure 1. Diagram of components of the Earth Engine Code Editor at code.earthengine.google.com.

Python Client Library

Authentication

- Authentication
- Install & import Python packages
- Image Collection
- Map
- Table

```
COLAB_AUTH_FLOW_CLOUD_PROJECT_FOR_API_CALLS = None

import ee
import google
import os

if COLAB_AUTH_FLOW_CLOUD_PROJECT_FOR_API_CALLS is None:
  print("Authenticating using Notebook auth...")
  if os.path.exists(ee.oauth.get_credentials_path()) is False:
    ee.Authenticate()
  else:
    print('\N{check mark} '
          'Previously created authentication credentials were found.')
  ee.Initialize()
else:
  print('Authenticating using Colab auth...')
  # Authenticate to populate Application Default Credentials in the Colab VM.
  google.colab.auth.authenticate_user()
  # Create credentials needed for accessing Earth Engine.
  credentials, auth_project_id = google.auth.default()
  # Initialize Earth Engine.
  ee.Initialize(credentials, project=COLAB_AUTH_FLOW_CLOUD_PROJECT_FOR_API_CALLS)
print('\N{check mark} Successfully initialized!')
```

```
Authenticating using Notebook auth...
✓ Previously created authentication credentials were found.
✓ Successfully initialized!
```

Install & import Python packages

dData

- Authentication
- Install & import Python packages
- Image Collection
- Map
- Table

```
[2]
import geemap
geemap.update_package()
```

```
Downloading https://github.com/gee-community/geemap/archive/master.zip ...
Unzipping geemap-master.zip ...
Data downloaded to: C:\Users\Norapath Arjanurak\Downloads\geemap-master

Please comment out 'geemap.update_package()' and restart the kernel to take effect:
Jupyter menu -> Kernel -> Restart & Clear Output
```

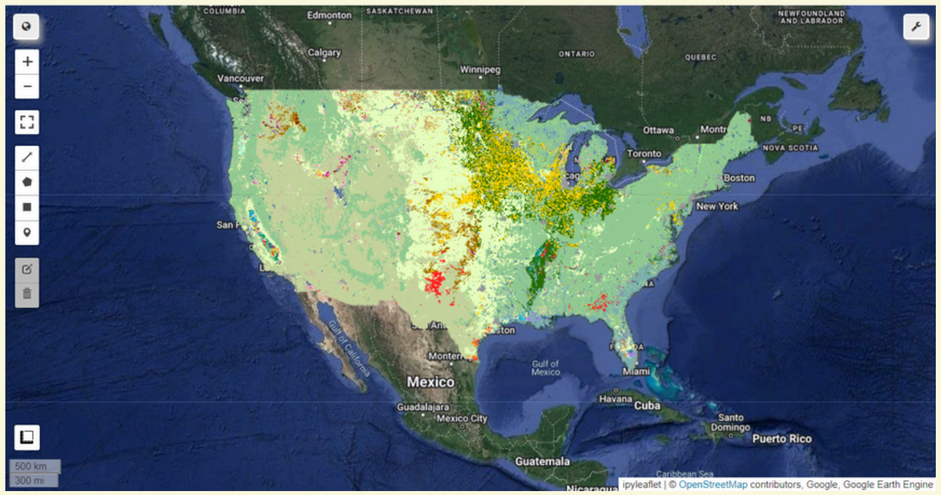
Image Collection

- Authentication
- Install & import Python packages
- Image Collection
- Map
- Table

```
USDAdata=ee.ImageCollection("USDA/NASS/CDL")
```

```
dataset = ee.ImageCollection('USDA/NASS/CDL').filter(ee.Filter.date('2018-01-01', '2019-12-31')).first()
cropLandcover = dataset.select('cropland')
Map1=geemap.Map(**default_center)
Map1.add_basemap('HYBRID')
Map1.addLayer(cropLandcover, {}, 'Crop Landcover')
Map1
```

[9] ✓ 1.9s Python



ee.ImageCollection.first

Returns the first entry from a given collection.

Image Collection-getInfo()

- Authentication
- Install & import Python packages
- Image Collection
- Map
- Table

```
dataset.getInfo()
```

✓ 1.6s


```
{'type': 'Image',
 'bands': [{'id': 'cropland',
   'data_type': {'type': 'PixelType',
    'precision': 'int',
    'min': 0,
    'max': 255},
   'dimensions': [153811, 96523],
   'crs': 'PROJCS["Albers Conical Equal Area", \n  GEOGCS["NAD83", \n    DATUM["North_American_Datum_1983", \n      SPHEROID["GRS 1980
   'crs_transform': [30, 0, -2356095, 0, -30, 3172605]},
  {'id': 'cultivated',
   'data_type': {'type': 'PixelType',
    'precision': 'int',
    'min': 0,
    'max': 255},
   'dimensions': [153811, 96523],
   'crs': 'PROJCS["Albers Conical Equal Area", \n  GEOGCS["NAD83", \n    DATUM["North_American_Datum_1983", \n      SPHEROID["GRS 1980
   'crs_transform': [30, 0, -2356095, 0, -30, 3172605]},
  {'id': 'confidence',
   'data_type': {'type': 'PixelType',
    'precision': 'int',
    'min': 0,
    'max': 255},
   'dimensions': [153811, 96523],
   'crs': 'PROJCS["Albers Conical Equal Area", \n  GEOGCS["NAD83", \n    DATUM["North_American_Datum_1983", \n      SPHEROID["GRS 1980
   'crs_transform': [30, 0, -2356095, 0, -30, 3172605]}],
...
   '',
   '',
   '',
   'Dbl Crop Barley/Soybeans'],
  'system:index': '2018'}}
```


Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Image Collection- 4 Separated dataset

dData

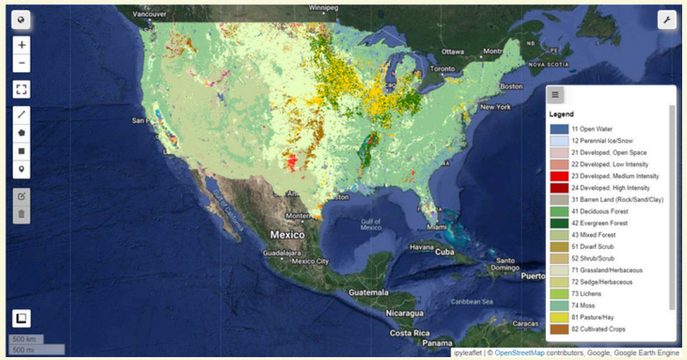
- Authentication
- Install & import Python packages
- Image Collection
- Map
- Table

```
dataset1 = ee.ImageCollection('USDA/NASS/CDL').filter(ee.Filter.date('2008-01-01', '2009-12-31')).first()
dataset2 = ee.ImageCollection('USDA/NASS/CDL').filter(ee.Filter.date('2010-01-01', '2013-12-31')).first()
dataset3 = ee.ImageCollection('USDA/NASS/CDL').filter(ee.Filter.date('2014-01-01', '2016-12-31')).first()
dataset4 = ee.ImageCollection('USDA/NASS/CDL').filter(ee.Filter.date('2017-01-01', '2019-12-31')).first()
```

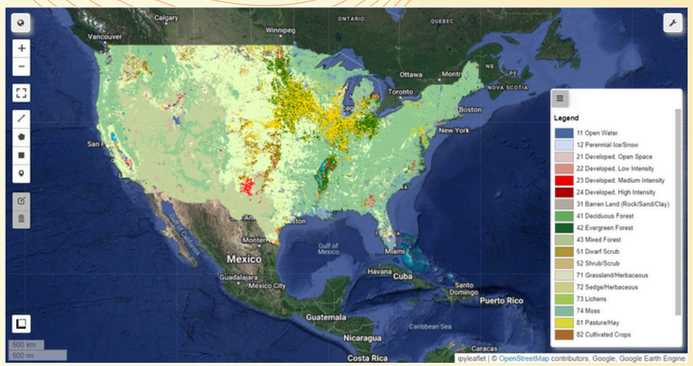
[31] ✓ 0.0s

Map- 4 Separated dataset

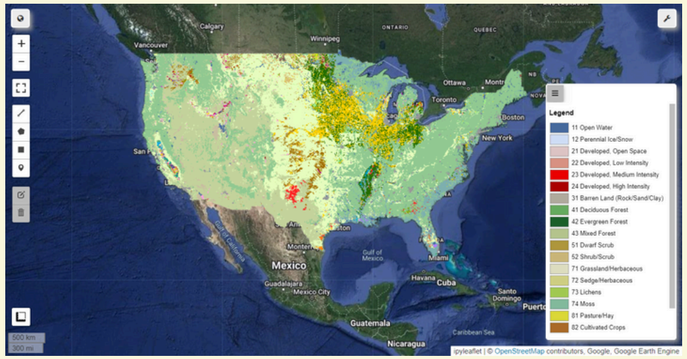
- Authentication
- Install & import Python packages
- Image Collection
- Map
- Table



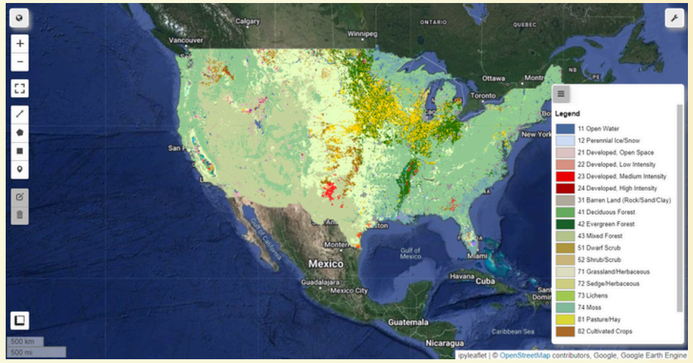
2008-2010



2011-2013



2014-2016



2017-2019

Image Collection- Group by state

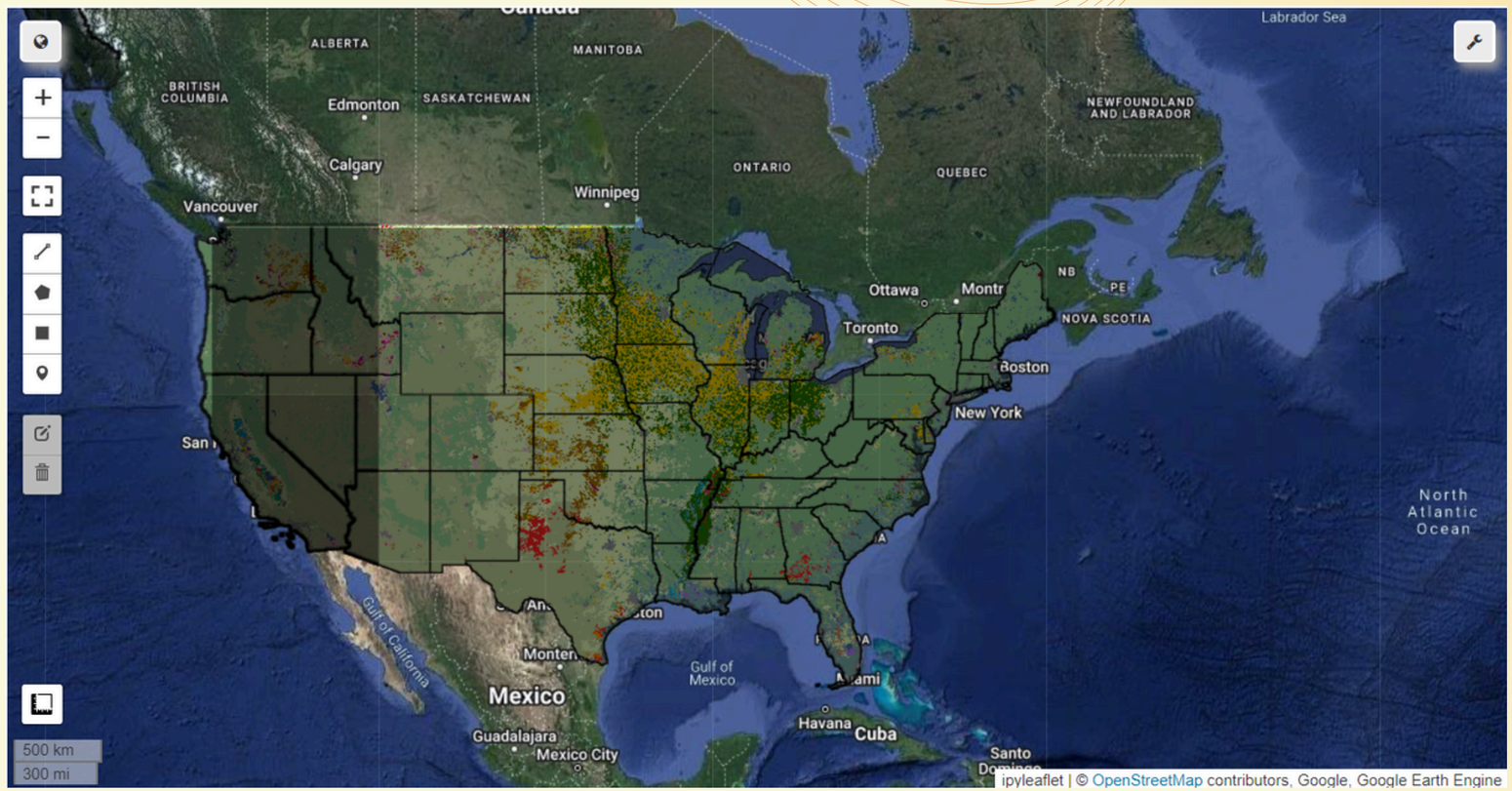
dData

- Authentication
- Install & import Python packages
- Image Collection
- Map
- Table

```
states = ee.FeatureCollection("TIGER/2018/States")
cropLandcover = dataset.select('cropland')
Map2=geemap.Map(**default_center)
Map2.add_basemap('HYBRID')
Map2.addLayer(cropLandcover, {}, 'Crop Landcover')
Map2.addLayer(states, {}, "US States")
Map2
```

Map- Group by state

- Authentication
- Install & import Python packages
- Image Collection
- Map
- Table



INTRODUNCTION | EDA | IMPLEMENT | RESULT | LESSON LEARN

Table- Group by state(Code)

dData

- Authentication
- Install & import Python packages
- Image Collection
- Map
- Table

```
out_dir = os.path.expanduser('~/Downloads')
out_dem_stats = os.path.join(out_dir, 'Crop_Mode_USA_1.csv')

if not os.path.exists(out_dir):
    os.makedirs(out_dir)

geemap.zonal_statistics(dataset1, states, out_dem_stats, statistics_type='MODE')
```

[23] Pythor

```
"\nout_dir = os.path.expanduser('~/Downloads')\nout_dem_stats = os.path.join(out_dir, 'Crop_Mode_USA_1.csv')\n\nif not os.path.exists
```

Table- Group by state

- Authentication
- Install & import Python packages
- Image Collection
- Map
- Table

cropland	cultivated	confidence	STATENS	GEOID	AWATER	LSAD	STUSPS	STATEFP	FUNCSTAT	INTPTLAT	DIVISION	REGION	NAME	INTPTLON	MTFCC	ALAND	system:index
141	1	78.4173	1219835	44	1.32E+09	0	RI	44	A	41.59742	1	1	Rhode Isla	-71.5273	G4000	2.68E+09	5
143	1	67.43058	1779794	33	1.03E+09	0	NH	33	A	43.67269	1	1	New Hamp	-71.5843	G4000	2.32E+10	7
141	1	69.37299	1779802	50	1.03E+09	0	VT	50	A	44.06858	1	1	Vermont	-72.6692	G4000	2.39E+10	9
141	1	88.3312	1779780	9	1.82E+09	0	CT	9	A	41.57986	1	1	Connecticu	-72.7467	G4000	1.25E+10	000000000000000000a
143	1	64.14429	1779787	23	1.17E+10	0	ME	23	A	45.40928	1	1	Maine	-68.6666	G4000	7.99E+10	25
141	1	70.41052	606926	25	7.13E+09	0	MA	25	A	42.15652	1	1	Massachus	-71.4896	G4000	2.02E+10	33
141	1	83.49693	1779795	34	3.54E+09	0	NJ	34	A	40.10727	2	1	New Jerse	-74.6652	G4000	1.9E+10	000000000000000000e
141	1	87.49822	1779798	42	3.39E+09	0	PA	42	A	40.9025	2	1	Pennsylvar	-77.8335	G4000	1.16E+11	12
141	1	74.47042	1779796	36	1.92E+10	0	NY	36	A	42.9134	2	1	New York	-75.5963	G4000	1.22E+11	26
1	2	100	1779784	17	6.21E+09	0	IL	17	A	40.10288	3	2	Illinois	-89.1526	G4000	1.44E+11	2
141	1	77.35242	1779806	55	2.93E+10	0	WI	55	A	44.63091	3	2	Wisconsin	-89.7094	G4000	1.4E+11	000000000000000000f
141	1	97.36386	1085497	39	1.03E+10	0	OH	39	A	40.41493	3	2	Ohio	-82.712	G4000	1.06E+11	18
141	1	66.44911	1779789	26	1.04E+11	0	MI	26	A	44.84418	3	2	Michigan	-85.6605	G4000	1.47E+11	000000000000000002a

Repeat 4 times!

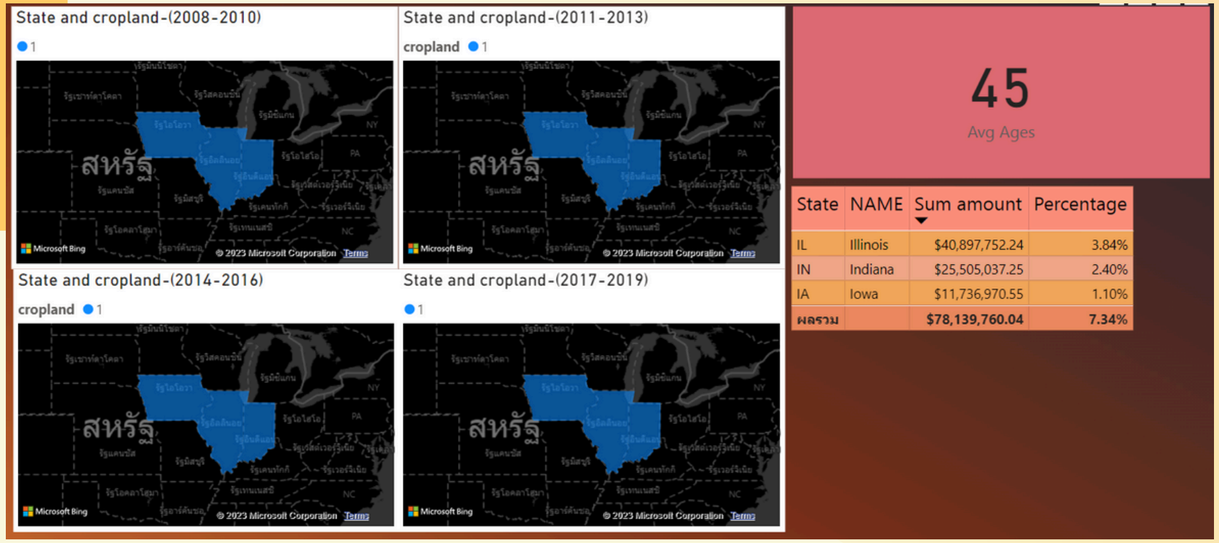
Result

Power BI

Result-Summarize#1

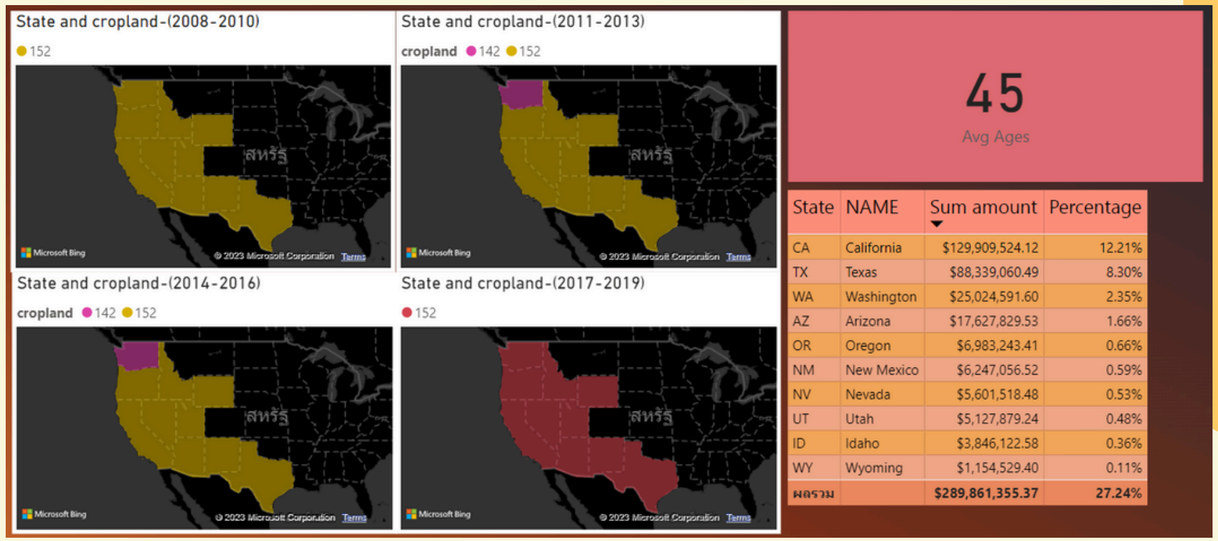
Corn

- Corn is the most popular crop around Illinois, Indiana, and Iowa with transaction in total of around 78 million dollars covered 7.34 percent.
- If the merchant want to buy the corn product, we recommend them to buy it around this area



State	NAME	Sum amount	Percentage
IL	Illinois	$40,897,752.24	3.84%
IN	Indiana	$25,505,037.25	2.40%
IA	Iowa	$11,736,970.55	1.10%
ผลรวม		$78,139,760.04	7.34%

Result-Summarize#2



45
Avg Ages

State	NAME	Sum amount	Percentage
CA	California	$129,909,524.12	12.21%
TX	Texas	$88,339,060.49	8.30%
WA	Washington	$25,024,591.60	2.35%
AZ	Arizona	$17,627,829.53	1.66%
OR	Oregon	$6,983,243.41	0.66%
NM	New Mexico	$6,247,056.52	0.59%
NV	Nevada	$5,601,518.48	0.53%
UT	Utah	$5,127,879.24	0.48%
ID	Idaho	$3,846,122.58	0.36%
WY	Wyoming	$1,154,529.40	0.11%
ผลรวม		$289,861,355.37	27.24%

Shrubland

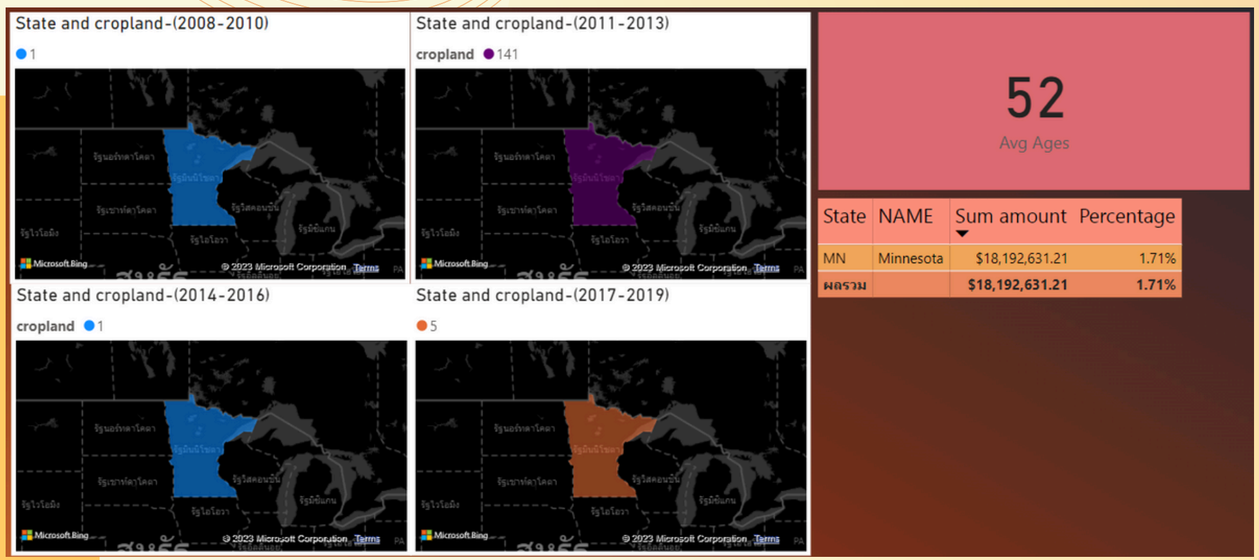
- Based on this data, it indicates that the western section of the USA is covered with shrubland(ทุ่งหญ้า), which they can use this area for real estate since it is the plane land.
- Moreover, the person in this area seems to have high spending. Investing in the high-end product seems to be good option.

Result-Summarize#3

Soybean

- Soybean is the newest crop in state plated in Minnesota.
- Soybeans are protein-rich and versatile, used in foods like tofu and animal feed, and play a role in biodiesel and industrial products. Their health benefits are studied, but their environmental impact highlights the need for sustainable farming practices.

ChatGBT



Summarize

1. Corn: Agriculture Product
2. Shrubland: Real-Estate
3. Soybean: Potential Product

Power BI

Lesson Learn

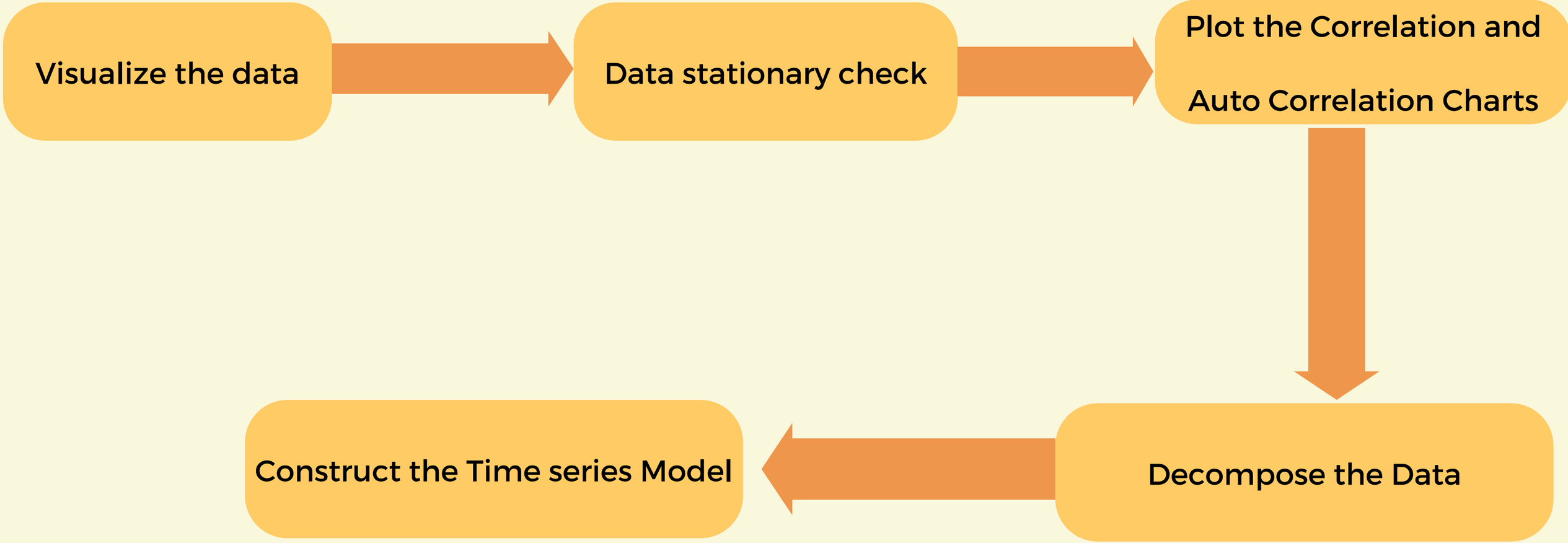
1. Google Earth Engine Experience
2. Python Environment
3. Library Support

Time Series Predictive Analytics Card Amount

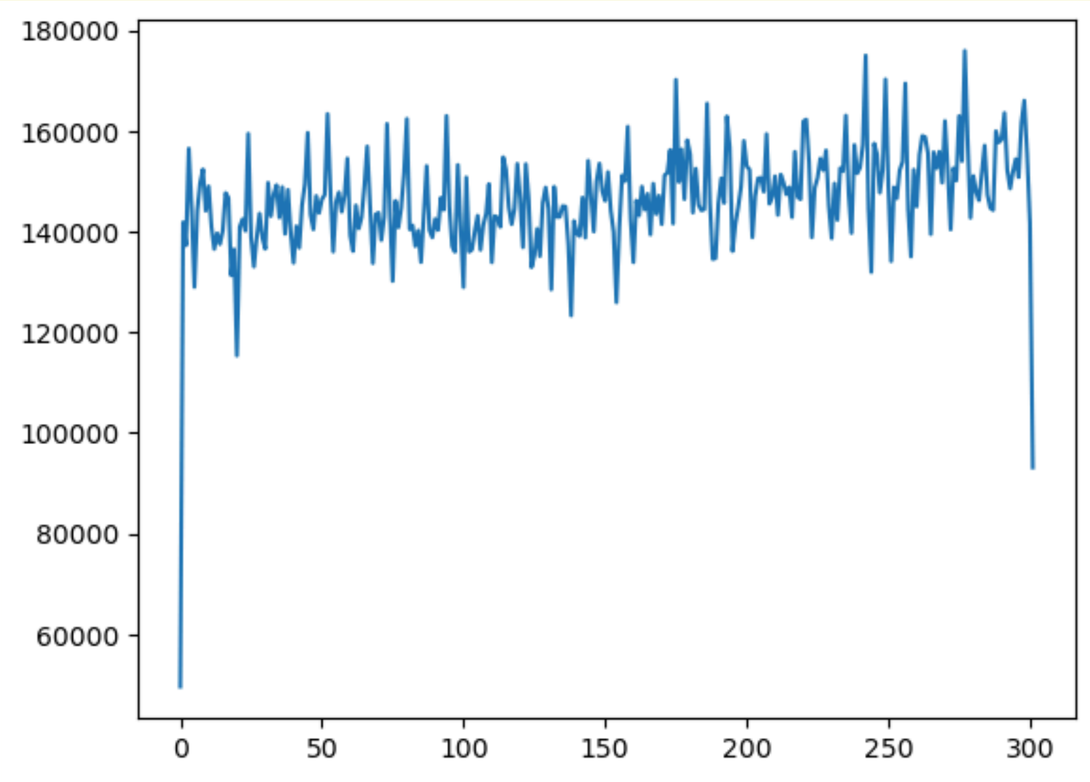
Transaction type amount prediction

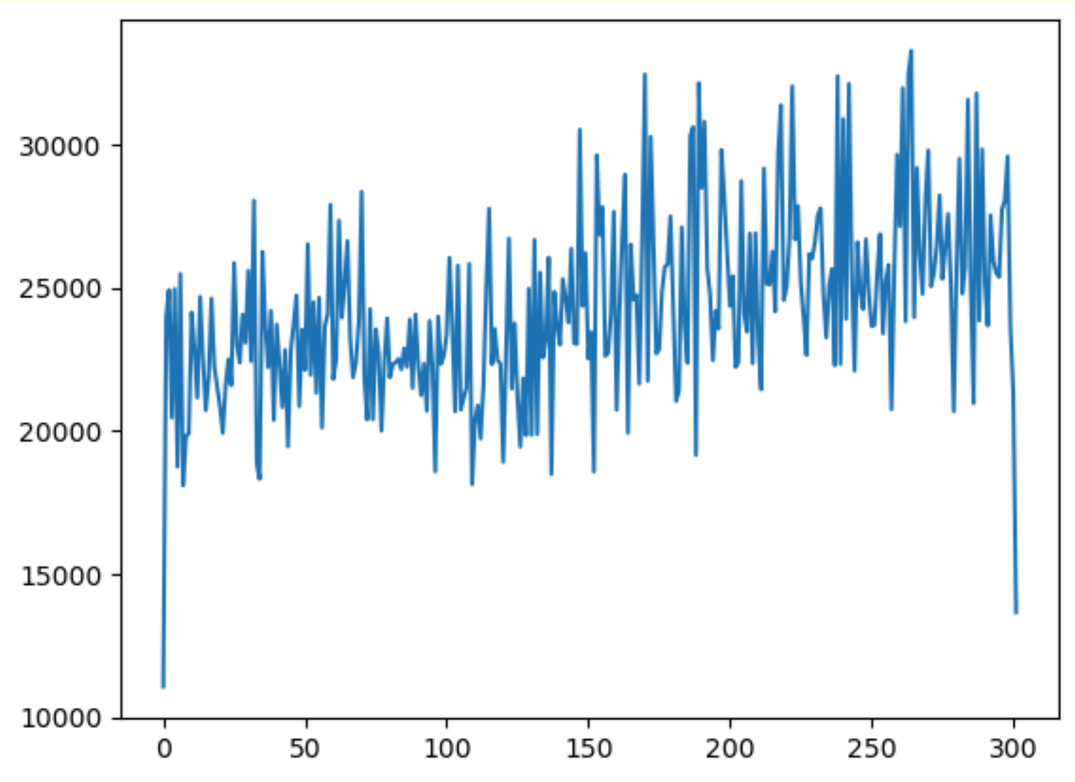
to forecast the daily total transaction amount by leveraging distinct credit card transaction types.

Model creation



Visualize the data





Data Stationary

In [44]:
```
def adfuller_test(sales):
    result=adfuller(sales)
    labels = ['ADF Test Statistic','p-value','#Lags Used','Number of Observations']
    for value,label in zip(result,labels):
        print(label+' : '+str(value) )

    if result[1] <= 0.05:
        print("strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data is stationary")
    else:
        print("weak evidence against null hypothesis,indicating it is non-stationary ")

adfuller_test(dfx1['Amount'])
```

```
ADF Test Statistic : -2.421820666170221
p-value : 0.1356721569203599
#Lags Used : 13
Number of Observations : 288
weak evidence against null hypothesis,indicating it is non-stationary
```

Data Stationary

In [45]:
```
dfx1['Sales First Difference'] = dfx1['Amount'] - dfx1['Amount'].shift(1)
dfx1['Seasonal First Difference']=dfx1['Amount']-dfx1['Amount'].shift(12)
dfx1.head()
```

Out[45]:

	Date	Amount	Sales First Difference	Seasonal First Difference
0	2009-08-16	49605.10	NaN	NaN
1	2009-08-17	141857.96	92252.86	NaN
2	2009-08-18	137257.46	-4600.50	NaN
3	2009-08-19	156505.96	19248.50	NaN
4	2009-08-20	144731.10	-11774.86	NaN

In [46]:
```
def adfuller_test(sales):
    result=adfuller(sales)
    labels = ['ADF Test Statistic','p-value','#Lags Used','Number of Observations']
    for value,label in zip(result,labels):
        print(label+' : '+str(value) )

adfuller_test(dfx1['Seasonal First Difference'].dropna())
```

```
ADF Test Statistic : -5.747776360989034
p-value : 6.064917205323806e-07
#Lags Used : 16
Number of Observations : 273
```

Data Stationary

In [84]:
```
def adfuller_test(sales):
    result=adfuller(sales)
    labels = ['ADF Test Statistic','p-value','#Lags Used','Number of Observations']
    for value,label in zip(result,labels):
        print(label+' : '+str(value) )

    if result[1] <= 0.05:
        print("strong evidence against the null hypothesis(Ho), reject the null hypothesis. Data is stationary")
    else:
        print("weak evidence against null hypothesis,indicating it is non-stationary ")

adfuller_test(dfx1['Amount'])
```

```
ADF Test Statistic : -2.7443145027093356
p-value : 0.06667879049687654
#Lags Used : 7
Number of Observations : 294
weak evidence against null hypothesis,indicating it is non-stationary
```

Data Stationary

In [85]:
```
dfx1['Sales First Difference'] = dfx1['Amount'] - dfx1['Amount'].shift(1)
dfx1['Seasonal First Difference']=dfx1['Amount']-dfx1['Amount'].shift(12)
dfx1.head()
```

Out[85]:

	Date	Amount	Sales First Difference	Seasonal First Difference
0	2009-08-16	11069.61	NaN	NaN
1	2009-08-17	23951.97	12882.36	NaN
2	2009-08-18	24908.36	956.39	NaN
3	2009-08-19	20456.03	-4452.33	NaN
4	2009-08-20	24946.96	4490.93	NaN

In [86]:
```
def adfuller_test(sales):
    result=adfuller(sales)
    labels = ['ADF Test Statistic','p-value','#Lags Used','Number of Observations']
    for value,label in zip(result,labels):
        print(label+' : '+str(value) )

adfuller_test(dfx1['Seasonal First Difference'].dropna())
```

```
ADF Test Statistic : -5.593712241922143
p-value : 1.3114500399060364e-06
#Lags Used : 15
Number of Observations : 274
```

Data Stationary

In [85]:
```
dfx1['Sales First Difference'] = dfx1['Amount'] - dfx1['Amount'].shift(1)
dfx1['Seasonal First Difference']=dfx1['Amount']-dfx1['Amount'].shift(12)
dfx1.head()
```

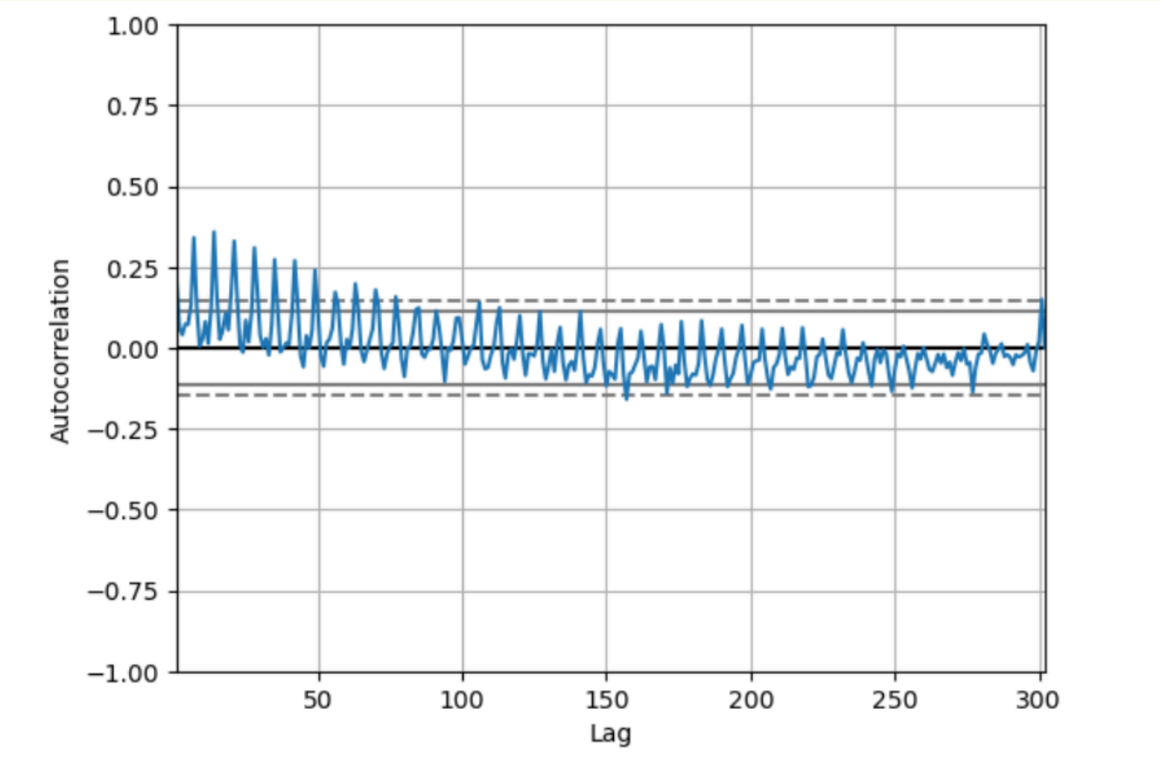
Out[85]:

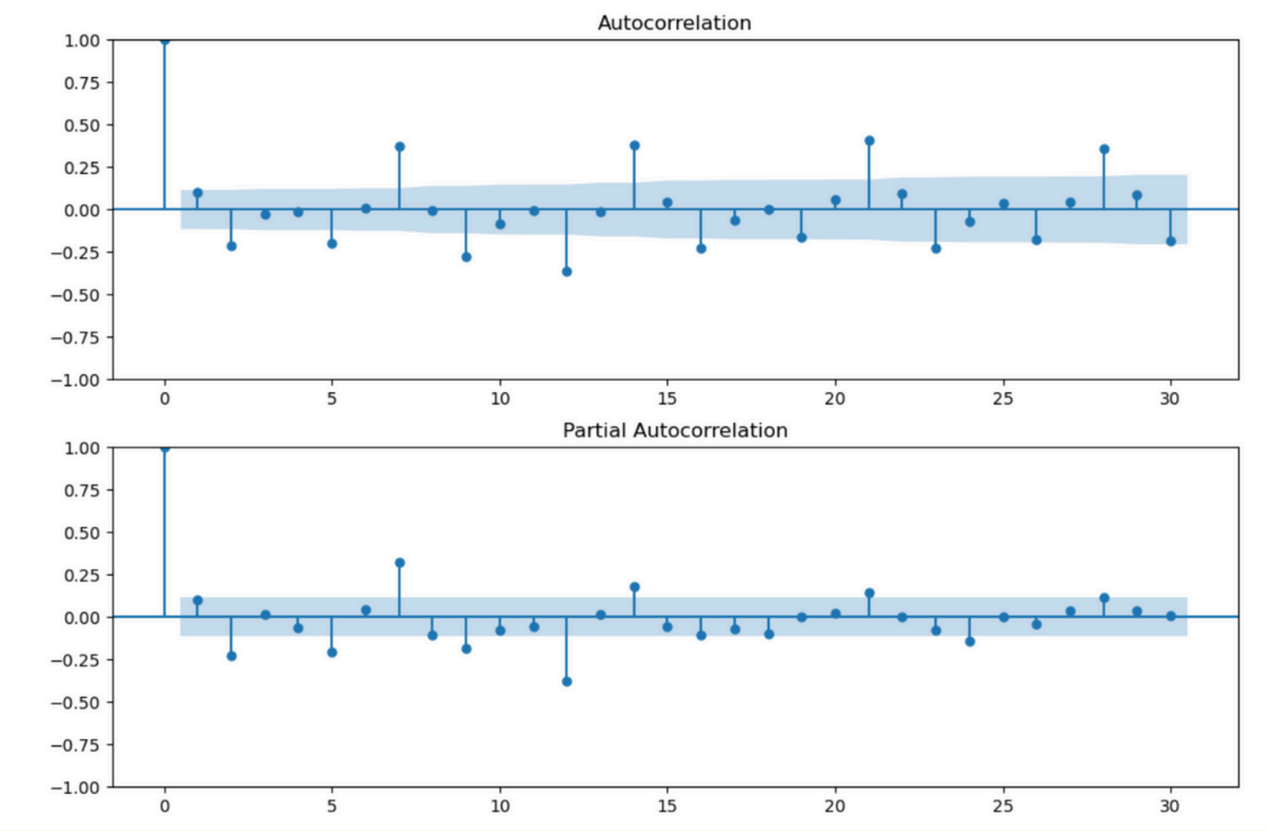
	Date	Amount	Sales First Difference	Seasonal First Difference
0	2009-08-16	11069.61	NaN	NaN
1	2009-08-17	23951.97	12882.36	NaN
2	2009-08-18	24908.36	956.39	NaN
3	2009-08-19	20456.03	-4452.33	NaN
4	2009-08-20	24946.96	4490.93	NaN

In [86]:
```
def adfuller_test(sales):
    result=adfuller(sales)
    labels = ['ADF Test Statistic','p-value','#Lags Used','Number of Observations']
    for value,label in zip(result,labels):
        print(label+' : '+str(value) )

adfuller_test(dfx1['Seasonal First Difference'].dropna())
```

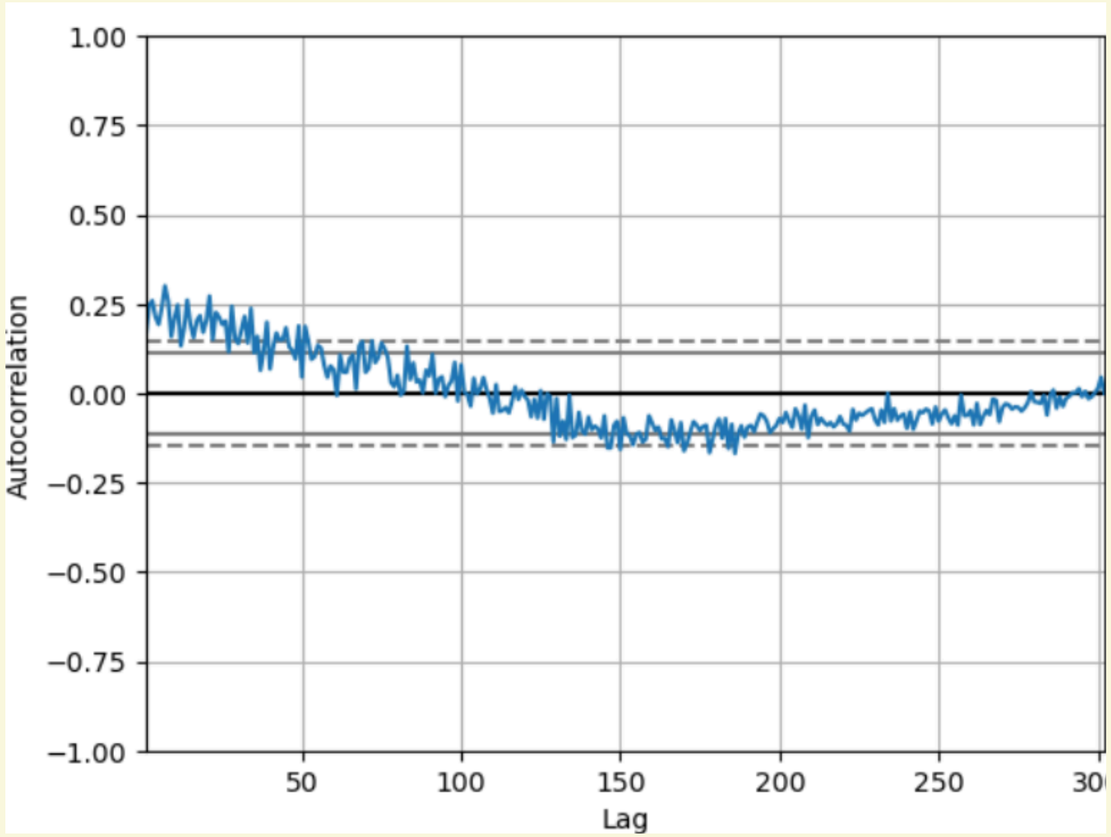
```
ADF Test Statistic : -5.593712241922143
p-value : 1.3114500399060364e-06
#Lags Used : 15
Number of Observations : 274
```

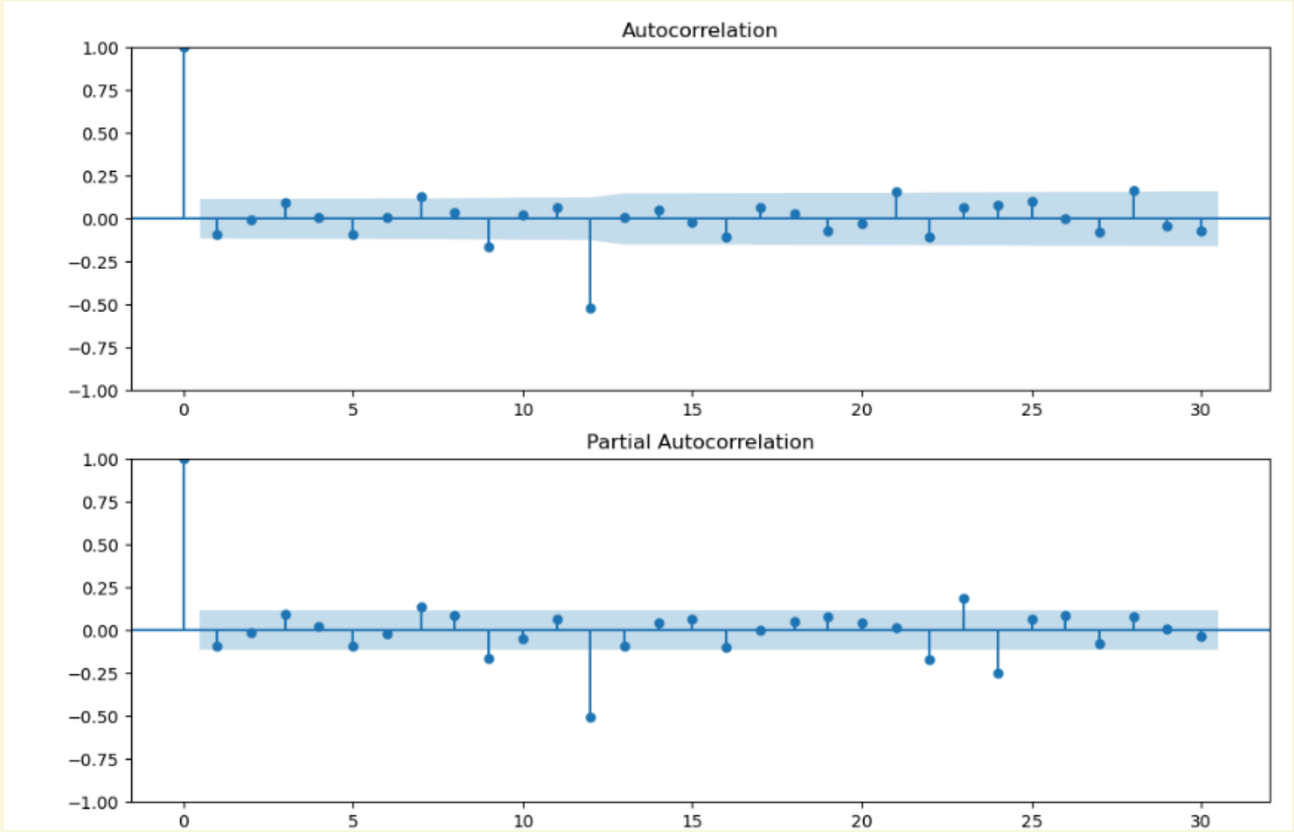
Plot the Correlation and Auto Correlation Charts



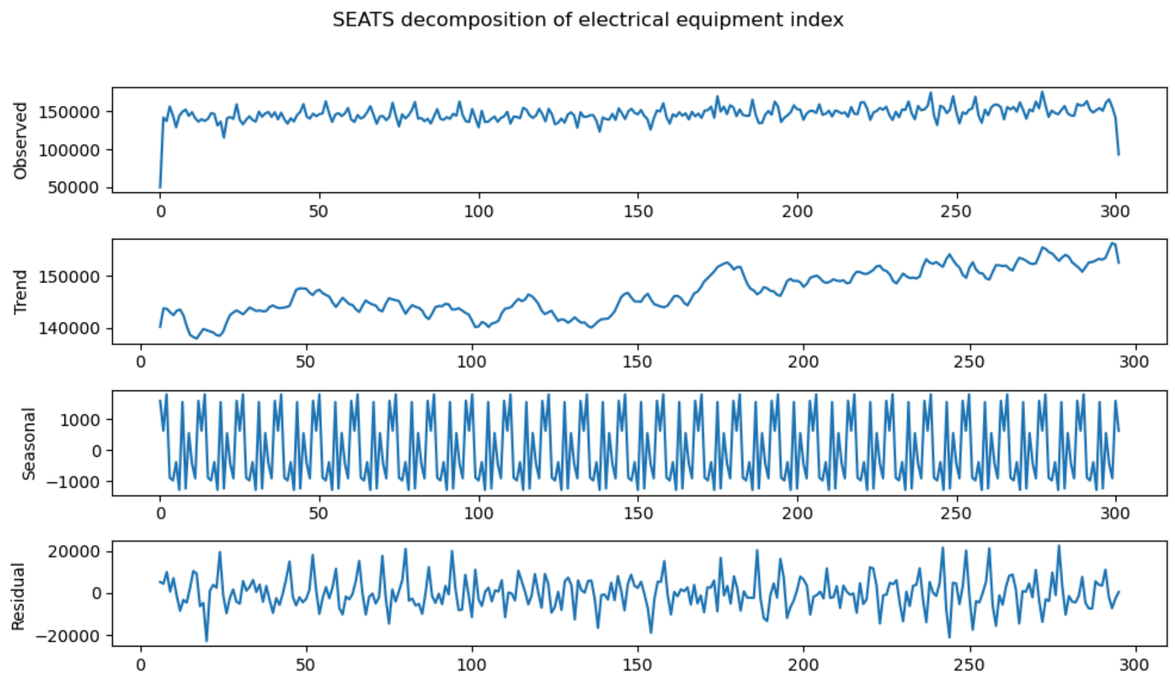


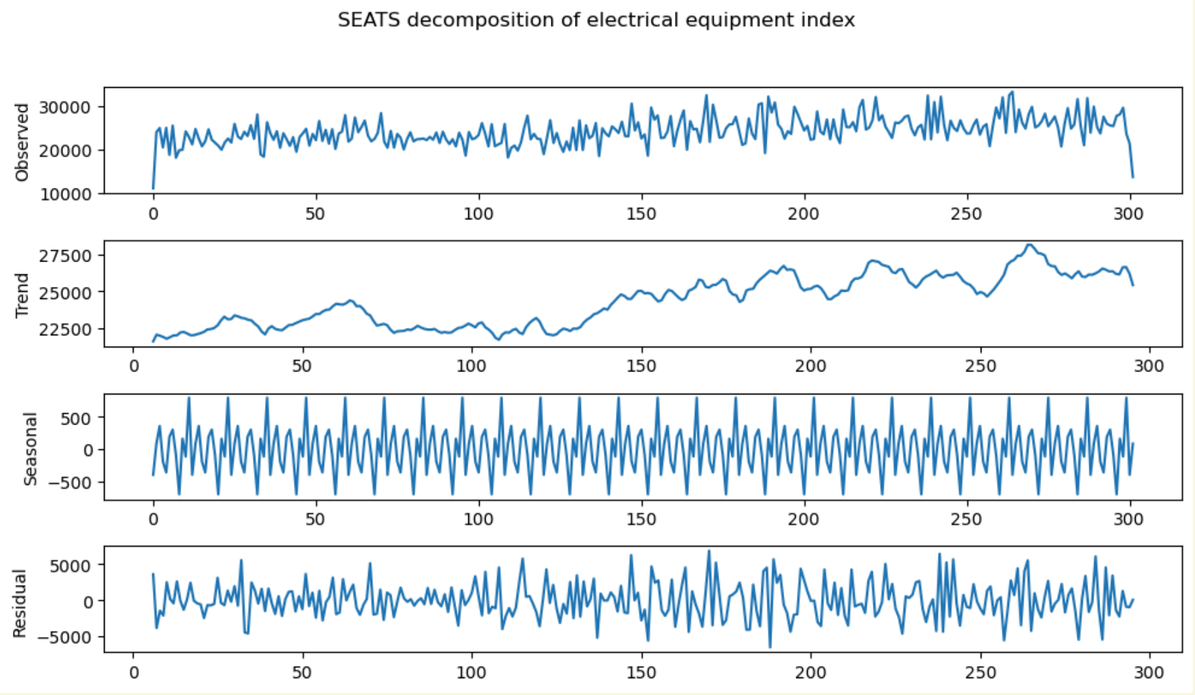
Plot the Correlation and Auto Correlation Charts





Decompose Data

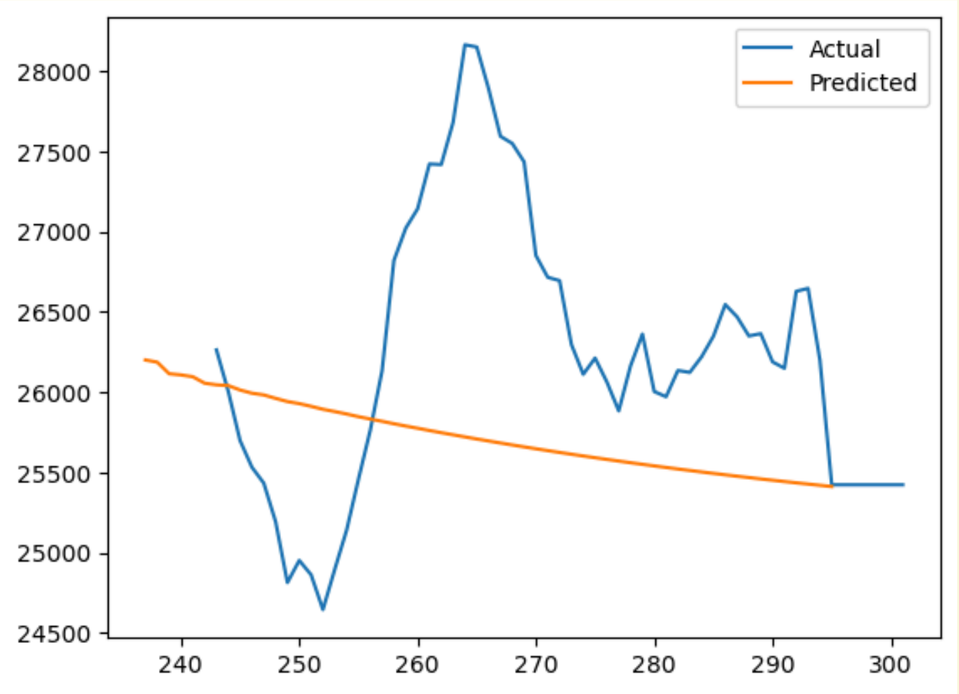




Construct the model

In [60]:
```
from statsmodels.tsa.api import AutoReg
#256
train_data=dfy[:round(0.8*len(dfy))]
test_data=dfy[round(0.8*len(dfy)):]
df_train = pd.DataFrame({'Amount': train_data})
df_test = pd.DataFrame({'Amount': test_data})
df = pd.DataFrame({'Amount': dfy})
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
lag = 7
ar_model = AutoReg(train_data, lags=lag).fit()

# Print Summary
print(ar_model.summary())
```

```
                            AutoReg Model Results
==============================================================================
Dep. Variable:                  trend   No. Observations:                  237
Model:                     AutoReg(7)   Log Likelihood               -1502.312
Method:               Conditional MLE   S.D. of innovations            166.143
Date:                Thu, 17 Aug 2023   AIC                           3022.623
Time:                        10:55:17   BIC                           3053.566
Sample:                             7   HQIC                          3035.105
                                  237
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const        251.8004    178.913      1.407      0.159     -98.863     602.464
trend.L1       1.7673      0.066     26.834      0.000       1.638       1.896
trend.L2      -1.4251      0.133    -10.708      0.000      -1.686      -1.164
trend.L3       1.2169      0.158      7.701      0.000       0.907       1.527
trend.L4      -0.9748      0.165     -5.907      0.000      -1.298      -0.651
trend.L5       0.5237      0.158      3.311      0.001       0.214       0.834
trend.L6      -0.1046      0.132     -0.794      0.427      -0.363       0.154
trend.L7      -0.0136      0.065     -0.209      0.834      -0.141       0.114
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -0.4557           -1.1558j            1.2424           -0.3098
AR.2           -0.4557           +1.1558j            1.2424            0.3098
AR.3            1.0180           -0.0000j            1.0180           -0.0000
AR.4            1.0235           -1.1148j            1.5134           -0.1318
AR.5            1.0235           +1.1148j            1.5134            0.1318
AR.6            1.7612           -0.0000j            1.7612           -0.0000
AR.7          -11.6163           -0.0000j           11.6163           -0.5000
-----------------------------------------------------------------------------
```



Construct the model

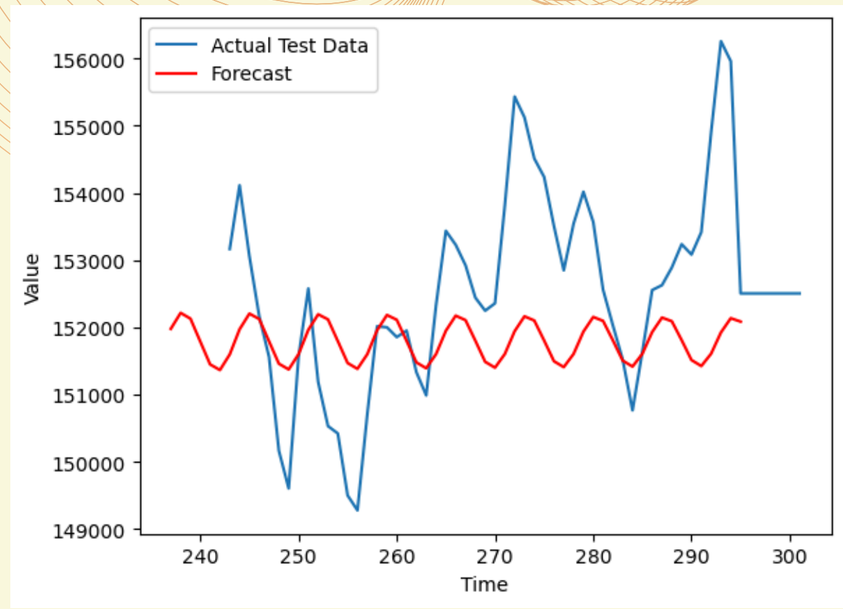
```
import numpy as np
import matplotlib.pyplot as plt
import statsmodels.api as sm
from statsmodels.tsa.arima.model import ARIMA
best_score = np.inf
best_order = (0, 0)
for p in range(0, 5):
    for q in range(0, 5):
        print('p =', p, 'q =', q)
        try:
            model = ARIMA(endog=train_data, order=(p, 1, q))  # Note the 'endog' argument
            model_fit = model.fit()
            aic = model_fit.aic
            if aic < best_score:
                best_score = aic
                best_order = (p, 1, q)
            print('AIC:', aic)
        except Exception as e:
            print('Error fitting model with p =', p, 'q =', q)
            print(e)

print('Best order (p, d, q):', best_order)
print('Best AIC:', best_score)
```

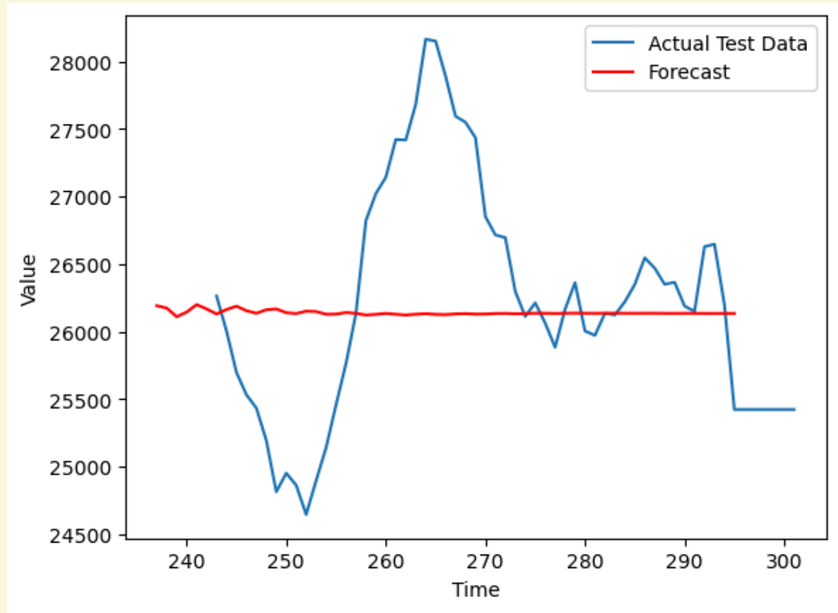
- The lowest AIC score for Swipe transaction is 3717.7820 which used (p,d,q) as (3,1,4)

- The lowest AIC score for Swipe transaction is 3053.2340 which used (p,d,q) as (3,1,4)

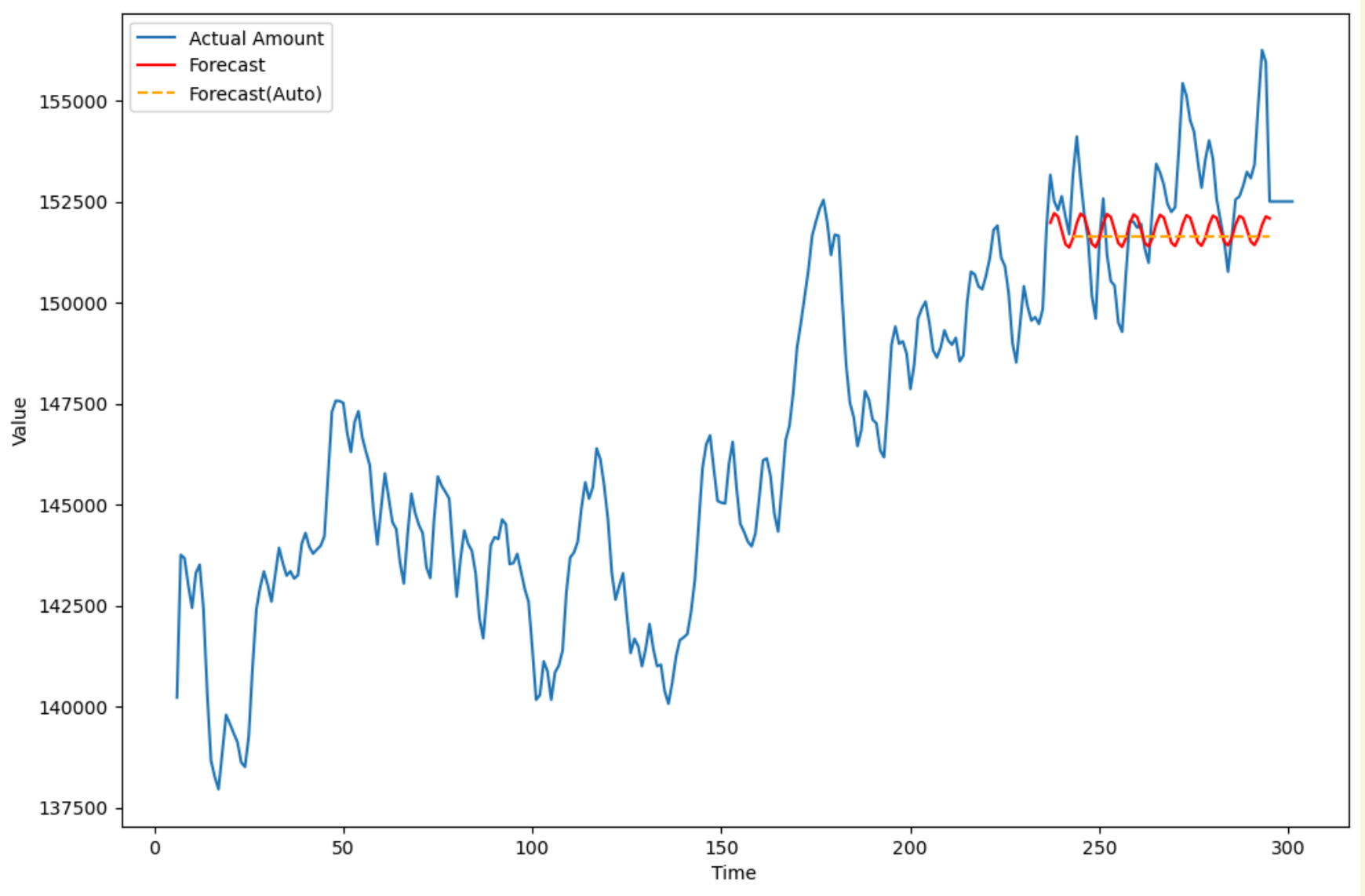
Construct the model

```
model = sm.tsa.ARIMA(train_data, order=(3,1,4))
result = model.fit()
model_fit = model.fit()
```

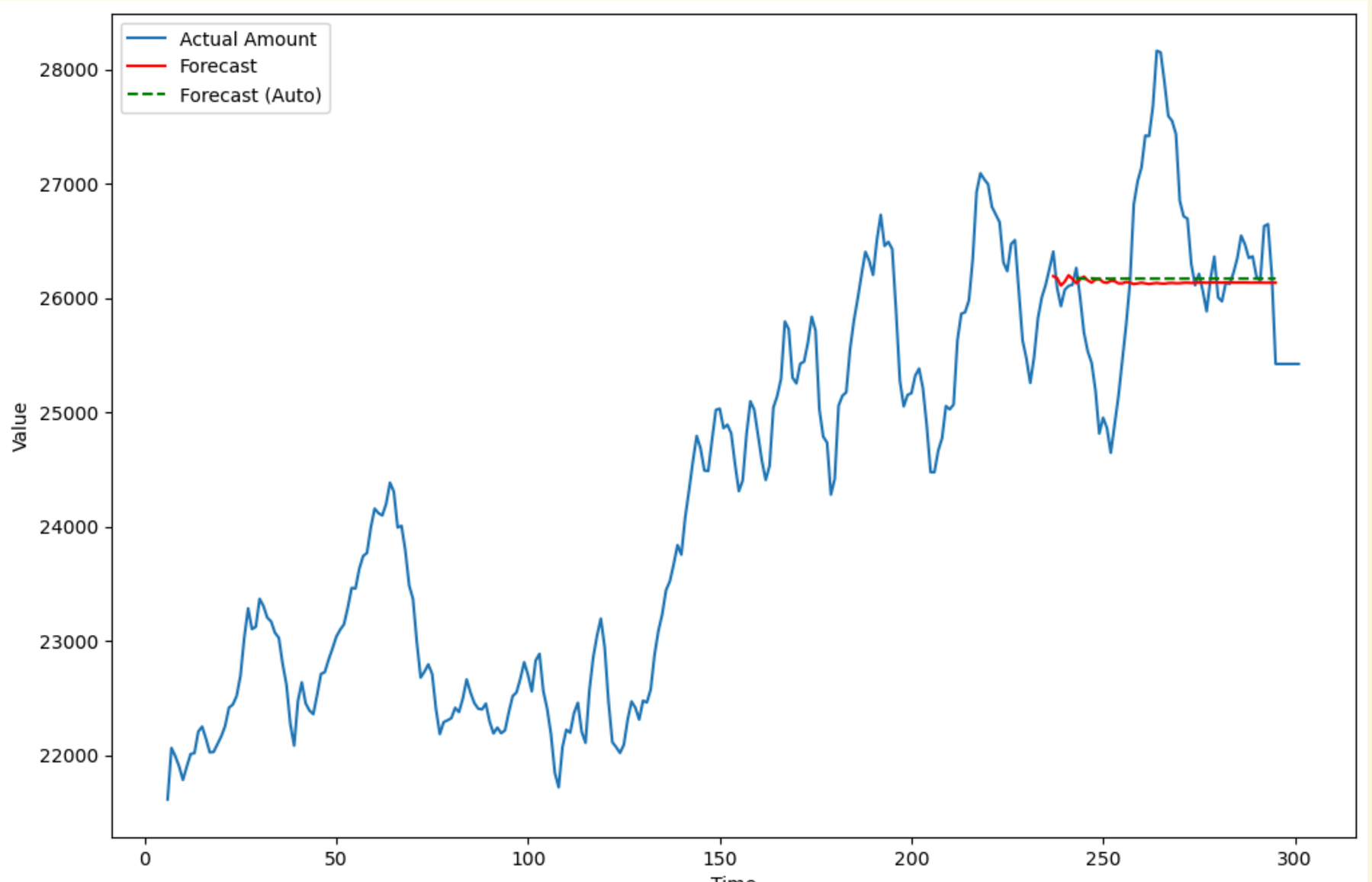




```
import pmdarima as pm
auto_arima = pm.auto_arima(train_data, stepwise=False, seasonal=False)
auto_arima
```



```
import pmdarima as pm
auto_arima = pm.auto_arima(train_data, stepwise=False, seasonal=False)
auto_arima
```



Construct the model

```python
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, mean_squared_error

mae = mean_absolute_error(test_data, forecast)
mape = mean_absolute_percentage_error(test_data, forecast)
rmse = np.sqrt(mean_squared_error(test_data, forecast))

print(f'mae - manual: {mae}')
print(f'mape - manual: {mape}')
print(f'rmse - manual: {rmse}')
```

```
mae - manual: 661.5452784711448
mape - manual: 0.025084330900089453
rmse - manual: 859.5545582949501
```

```
mae - auto: 655.8758487255512
mape - auto: 0.024895847772749488
rmse - auto: 850.390094724808
```

Construct the model

```python
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, mean_squared_error

mae = mean_absolute_error(test_data, forecast)
mape = mean_absolute_percentage_error(test_data, forecast)
rmse = np.sqrt(mean_squared_error(test_data, forecast))

print(f'mae - manual: {mae}')
print(f'mape - manual: {mape}')
print(f'rmse - manual: {rmse}')
```

```
mae - manual: 1232.328206913077
mape - manual: 0.008044566261815263
rmse - manual: 1552.2247509802417
```

```
mae - auto: 1369.1146091410794
mape - auto: 0.008934600483337926
rmse - auto: 1715.3061935173266
```

THANK YOU

6438121221@student.chula.ac.th
643822132@student.chula.ac.th