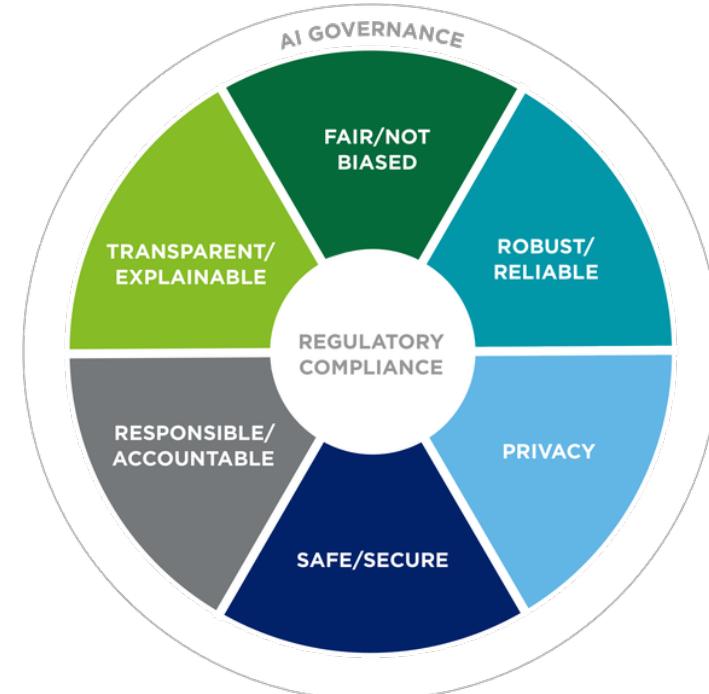




# Group E5

## Capstone Project

A requisite submission for the completion of Deloitte AI Academy™



# Meet the Team.



**Chandrasekhar  
Gudipati**



**Sushant  
Mudalgi**



**Shruti  
Maigur**



**Sushanth  
Ganesh**

# Problem Statement.

Let's see how various parameters affect loan default.



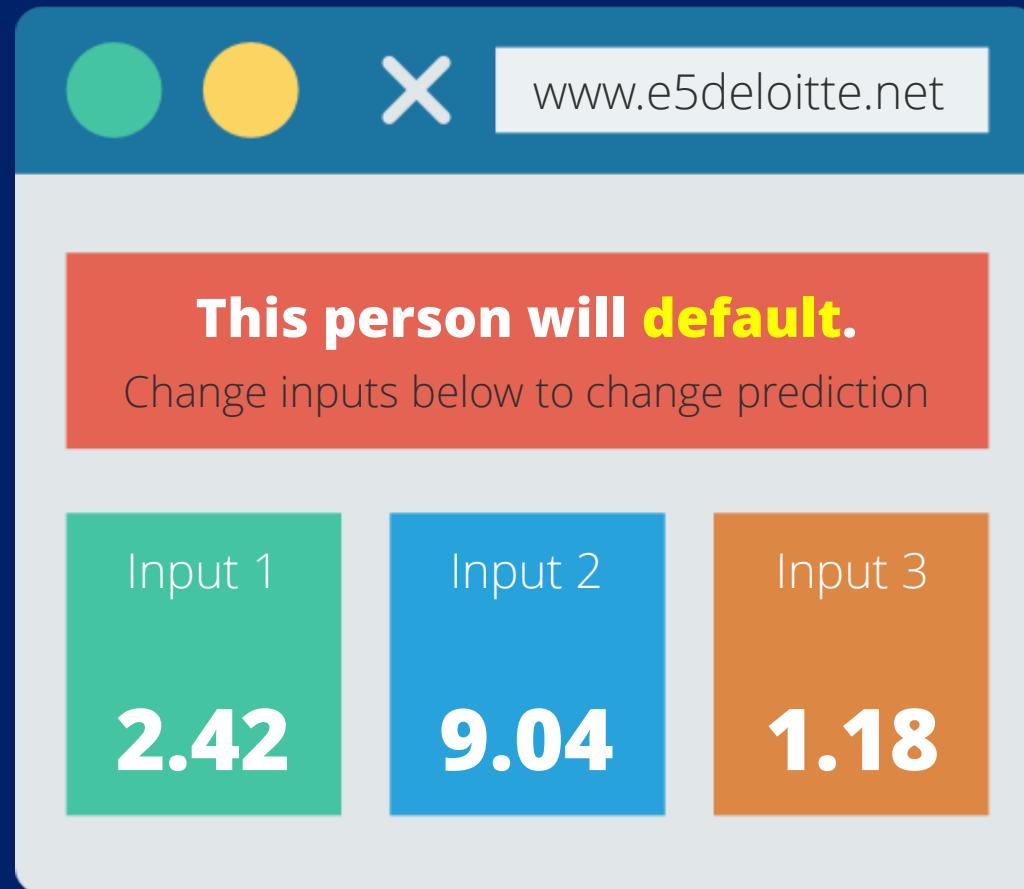
# Problem Statement.

Hey, can you give me  
a loan with the  
following details,  
Lorem Ipsum...



Let me check with  
the software brought  
to us by the folks at  
Team E5

# Problem Statement.



# Individual Data Files.

**Loan\_details.csv**



**Branch\_region\_  
mapping.csv**



**Loan\_status.csv**



# Individual Data Files.



**total\_table.csv**

# Combined Data.

Loan_id	disbursed_amount	asset_cost	ltv	branch_id	Date.of.Birth	Employment.Type	Disbursal.Date	MobileNo_Avl_Flag	Aadhar_flag
PAN_flag	VoterID_flag	Driving_flag	Passport_flag	PERFORM_CNS.SCORE	DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS	CREDIT.HISTORY.LENGTH	NO.OF_INQUIRIES	loan_default	region

# Work Methodology.



**Limited  
Time**



**Unlimited  
Exploration Possibilities**

# Work Methodology.



Waterfall vs Agile

# Work Methodology.



# Work Methodology.



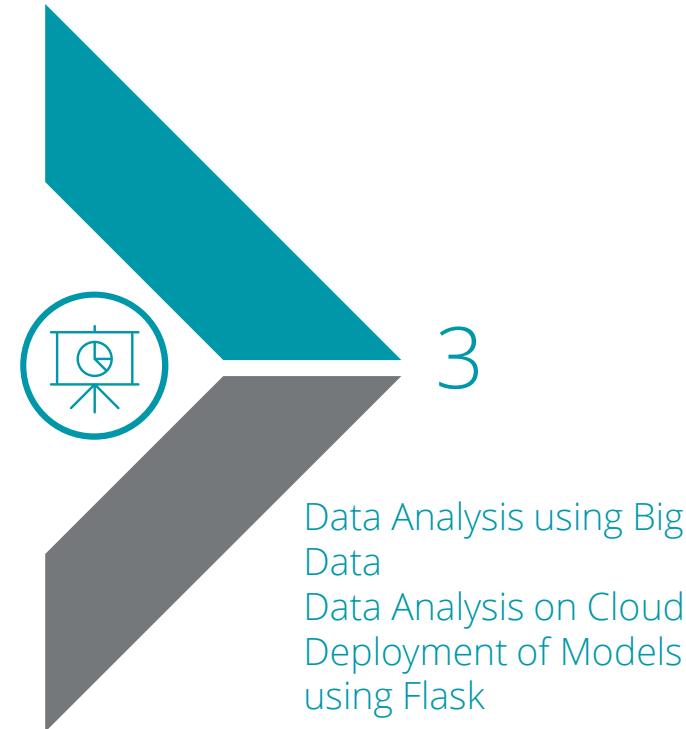
- **Two meetings were held each week.**
- **Once one member presented his/her work, others gave constructive criticism.**
- **This feedback was used to improve results.**

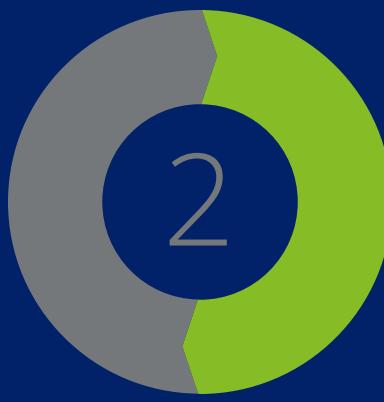
# So we'll let you in on our little Masterplan.



- 01 Exploration and Basic Visualization  
Poke the bear by going through the data and identifying patterns and correlations using visualizations and plots
- 02 Interactive Visualizations  
Create Tableau and PowerBI Dashboards that allow for interactive visualizations
- 03 PreProcessing  
Preprocess the data based on the inferences made
- 04 AI/ML  
Create various AI models based on the inferences and select the best of the lot
- 05 Data Engineering & Cloud  
Upload them to the cloud for the world to enjoy

# RoadMap.





18 October 2021

1

2

3

# Milestone One

# Individual Contributions.

1  
2  
3

Chandrasekhar Gudipati

1.1, PowerPoint

Shruti Maigur

1.3

Sushant Mudalgi

1.2

Sushanth Ganesh

1.2



1.1  
Data Manipulation  
using Python



1.2  
SQL



1.3  
Statistical Analysis  
using Python



1.1  
Data Manipulation  
using Python



1.2  
SQL



1.3  
Statistical Analysis  
using Python

Chandrasekhar Gudipati

1

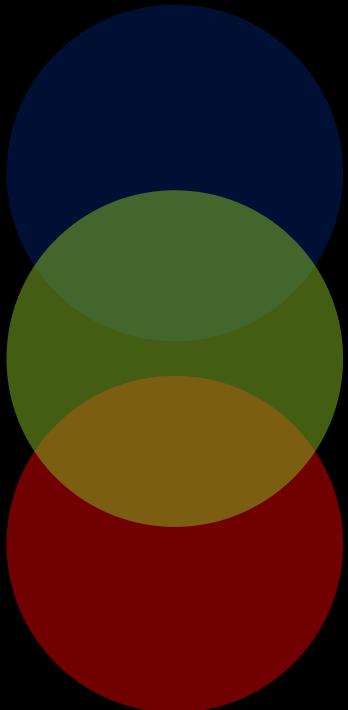
2

3

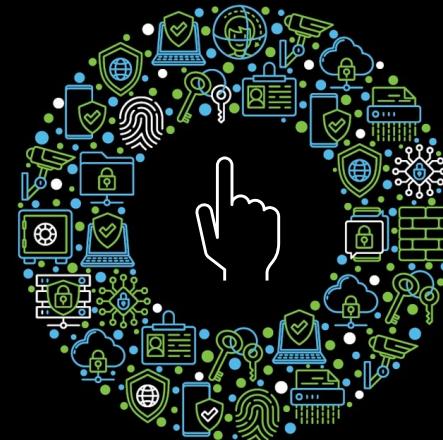
# 1.1 Data Manipulation using Python

# Table Merging.

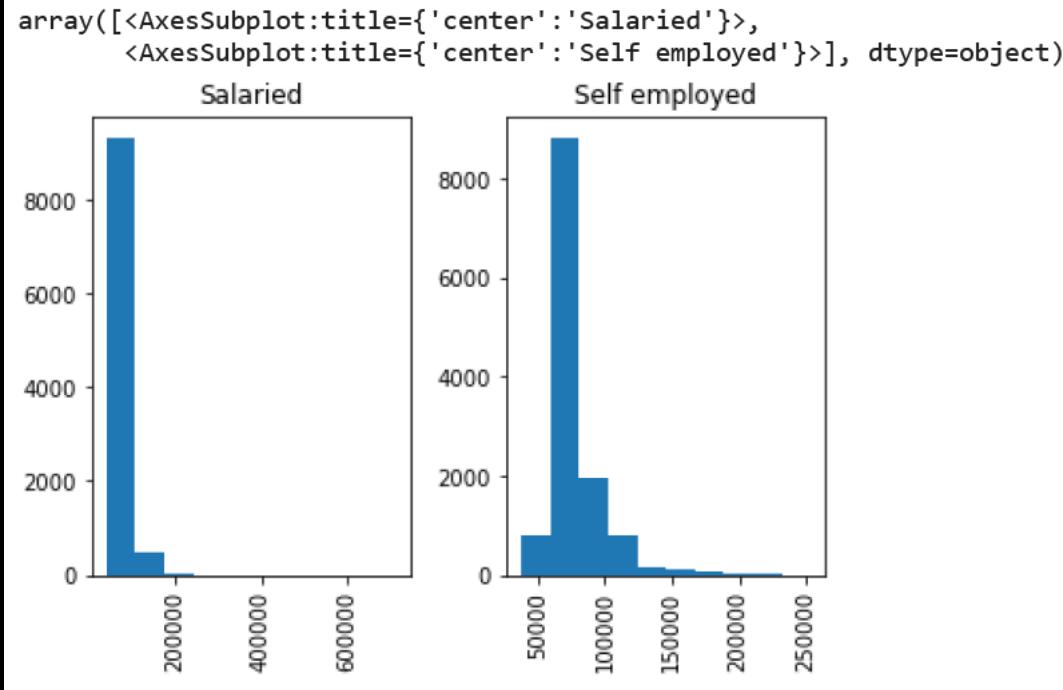
1  
2  
3



To begin with, we combined the data of the three given csv files into one pandas dataframe called `total_table`. Here's how.

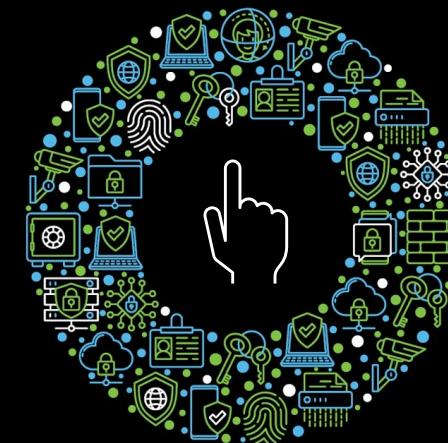


# Looking Around.

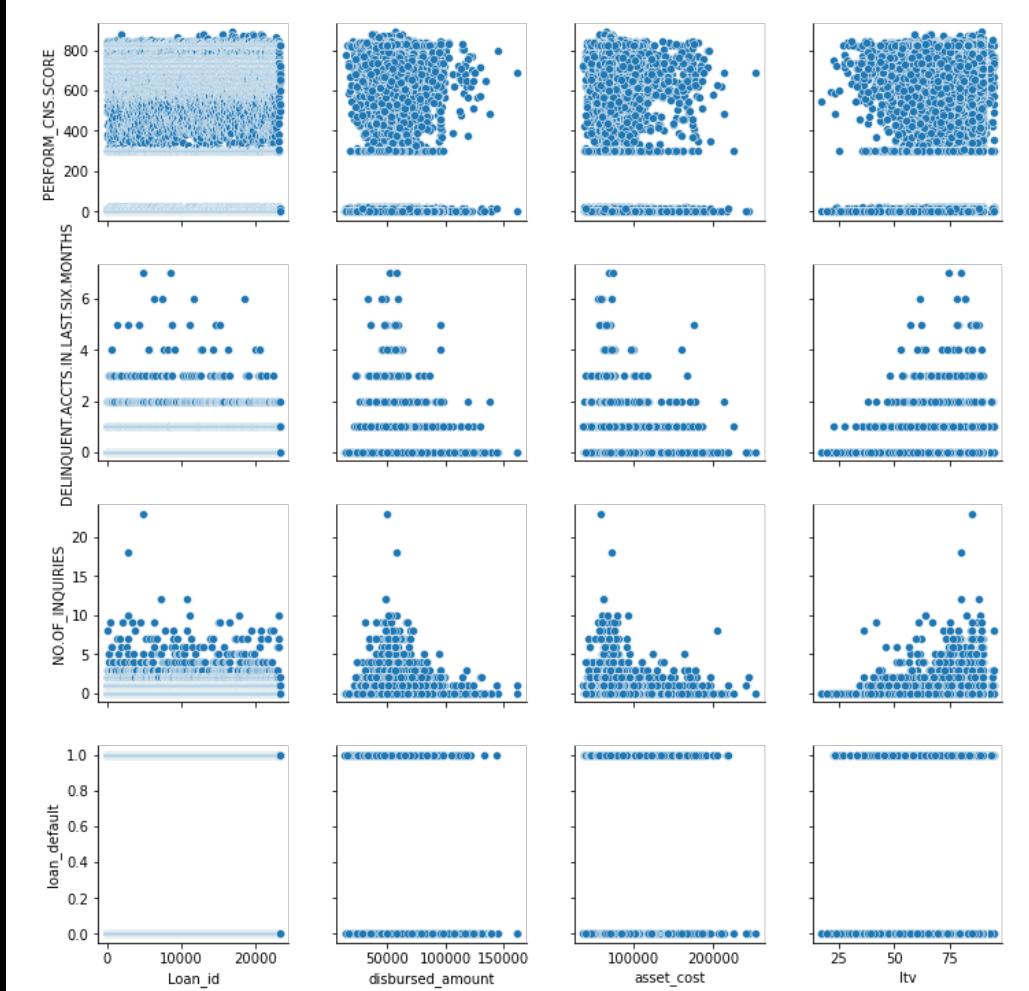


Mean, Median, Mode

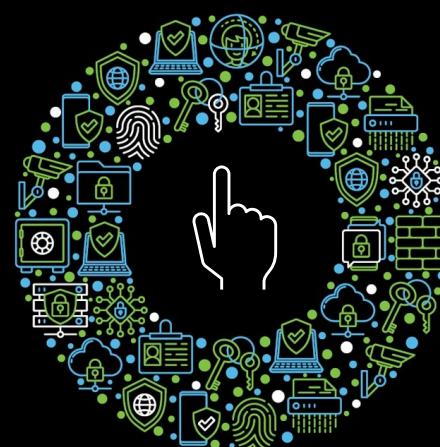
Then we began multiple analyses as alongside.  
Click to begin.



# Outliers Outliers Everywhere.



Handling outliers is a job of perspective. One person's outlier is another's gold mine. Click to know more.

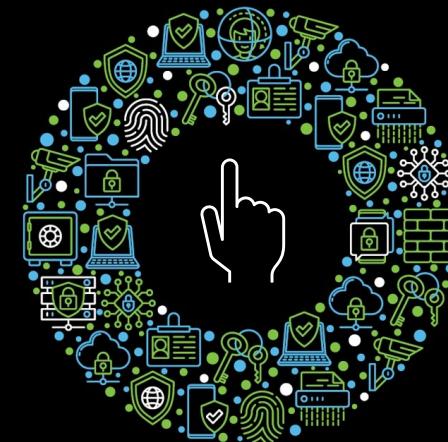


# Outliers Outliers Everywhere.



Asset Cost (X) vs Disbursed Amount(Y)

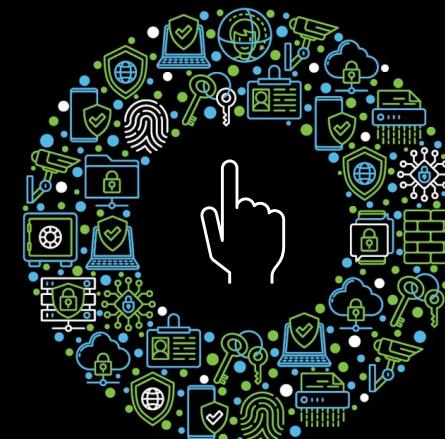
Both these outliers belong to bank 138. So guess where we started our detective work? Click to know more.



# Reading between the Lines.

	Loan_id	disbursed_amount	asset_cost	ltv	branch_id	Aadhar_flag	PAN_flag
Loan_id	1.000000	-0.006935	-0.003720	-0.005458	0.001564	0.004079	0.002083
disbursed_amount	-0.006935	1.000000	0.752629	0.372385	0.024890	-0.017392	0.013246
asset_cost	-0.003720	0.752629	1.000000	-0.304288	0.023997	-0.089512	0.043821
ltv	-0.005458	0.372385	-0.304288	1.000000	0.010844	0.098845	-0.032283
branch_id	0.001564	0.024890	0.023997	0.010844	1.000000	-0.041416	0.026347
Aadhar_flag	0.004079	-0.017392	-0.089512	0.098845	-0.041416	1.000000	-0.188042
PAN_flag	0.002083	0.013246	0.043821	-0.032283	0.026347	-0.188042	1.000000
VoterID_flag	-0.004142	0.016849	0.084220	-0.091096	0.025477	-0.862401	0.164873
Driving_flag	-0.001820	-0.001977	0.015421	-0.021724	-0.014848	-0.287130	0.000230
Passport_flag	-0.004154	0.001242	-0.005971	0.010548	-0.011556	-0.082545	-0.002240
PERFORM_CNS.SCORE	0.003165	0.009414	-0.052328	0.088537	-0.016329	0.067788	0.022877
DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS	-0.002406	0.020351	-0.002257	0.038652	-0.004391	0.022699	-0.000499
NO.OF_INQUIRIES	-0.005668	0.035428	-0.018974	0.087960	0.000974	0.028507	0.024491
loan_default	-0.010647	0.062805	0.002535	0.095423	0.037361	-0.050624	0.007771

We love the vibrant colours on this one. Click below for a chromatic feast. We understood a lot of patterns through these correlation plots.





1.1  
Data Manipulation  
using Python



1.2  
SQL



1.3  
Statistical Analysis  
using Python



1.1  
Data Manipulation  
using Python



1.2  
SQL



1.3  
Statistical Analysis  
using Python

Sushant Mudalgi, Sushanth Ganesh

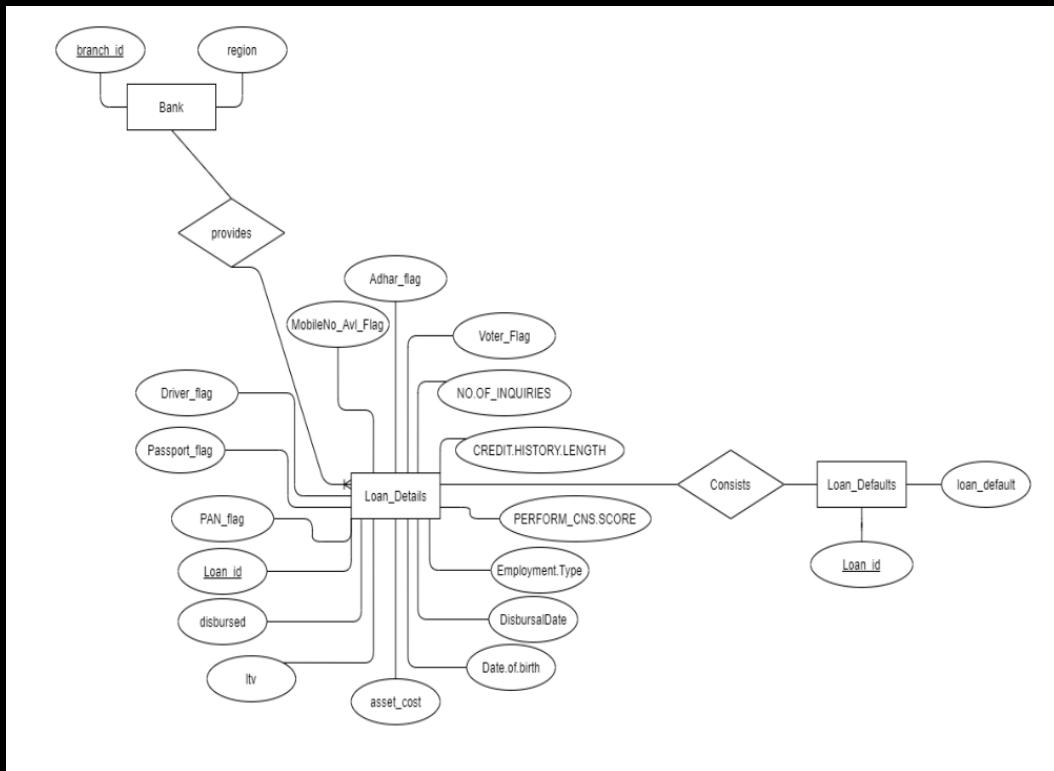
1

2

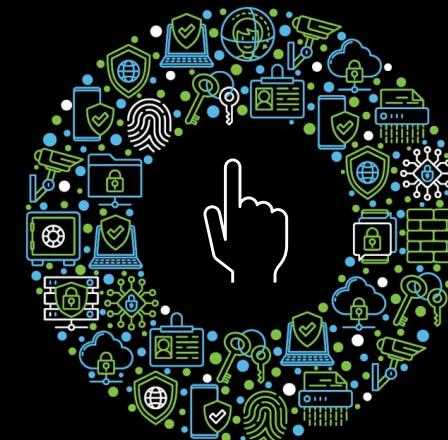
3

# 1.2 SQL

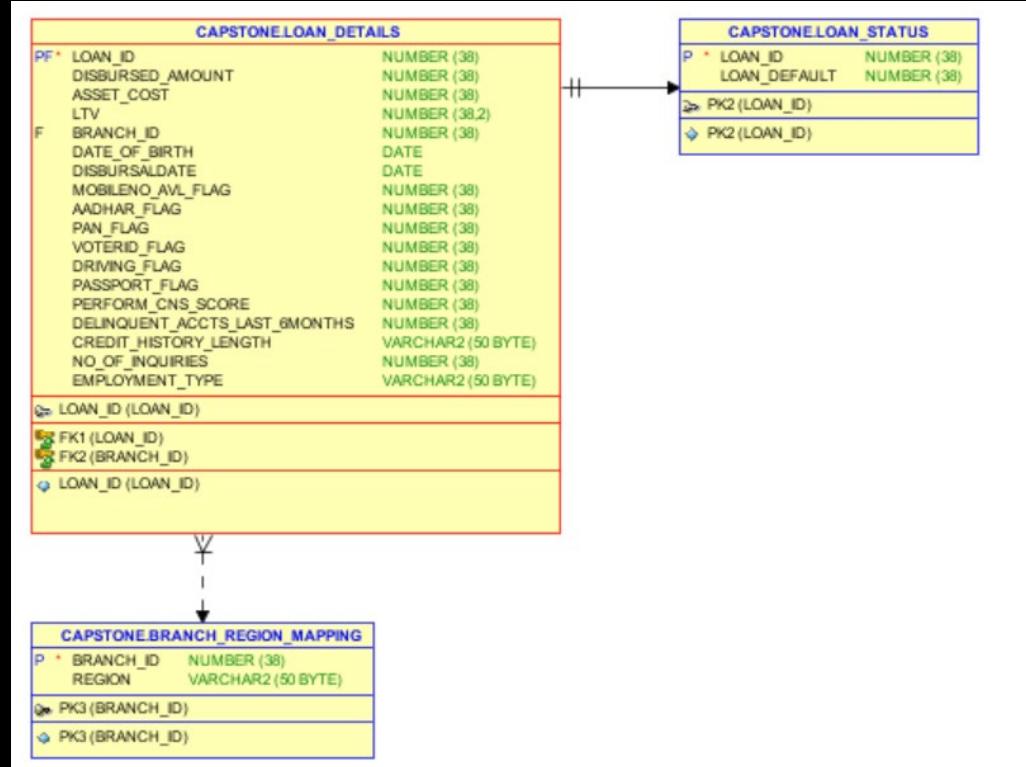
# ER Diagram.



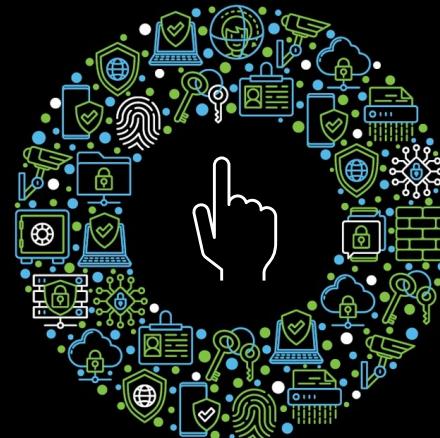
We started with an ER Diagram to get a basic understanding of our work. Click to begin.



# Functioning.



This was generated from SQL Developer tool after we created the table and assigned the keys. Click to proceed.





1.1  
Data Manipulation  
using Python



1.2  
SQL



1.3  
Statistical Analysis  
using Python



1.1  
Data Manipulation  
using Python



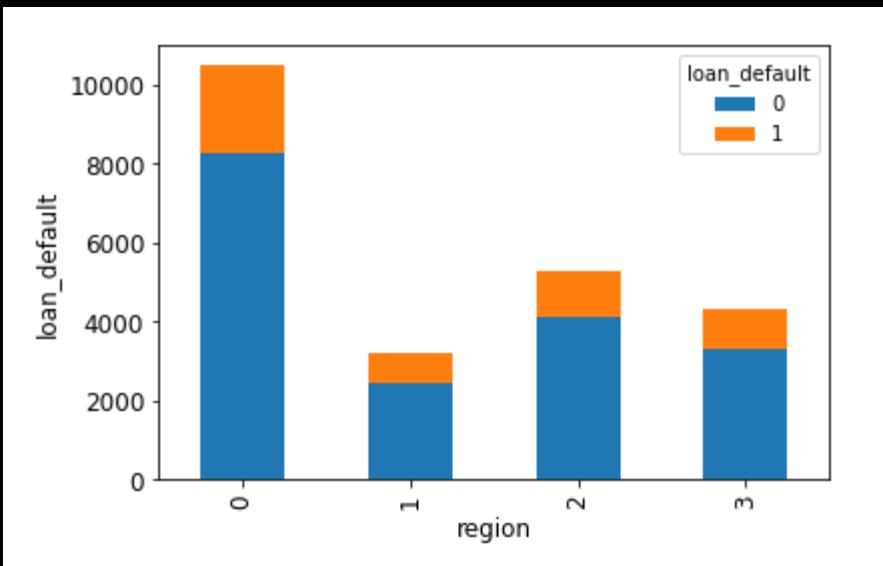
1.2  
SQL



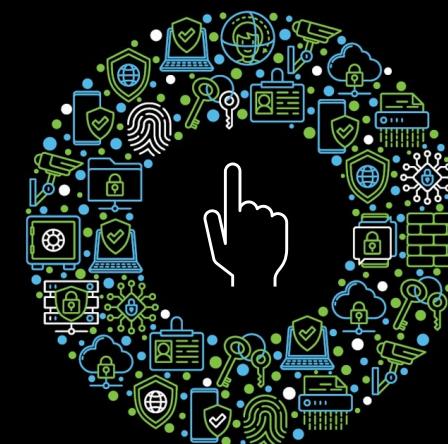
1.3  
Statistical Analysis  
using Python

# 1.3 Statistical Analysis using Python

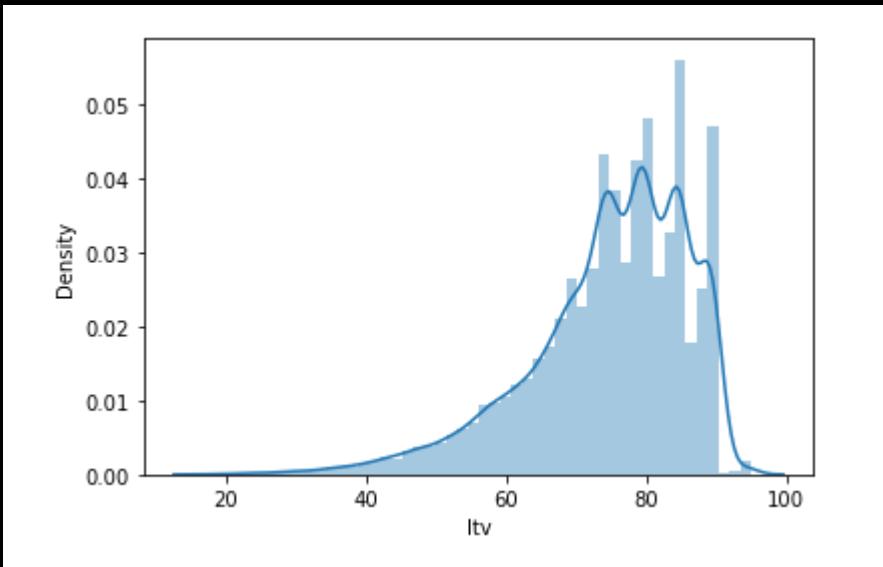
# Descriptive Statistics.



To understand the data just a little more, we began descriptive statistical analyses. Click to know more.



# Hypothesis Testing.

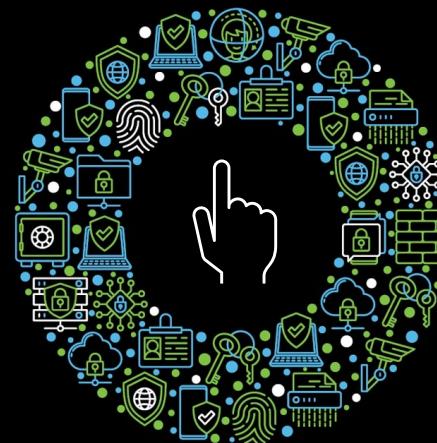


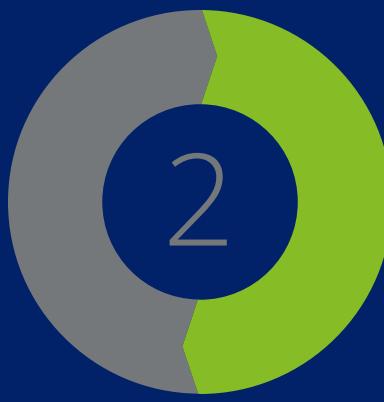
```
1 Data['ltv'].skew()
```

```
-1.0760494824413254
```

- Data is not normally distributed.
- Applying T-test on mean on pop and sample,
- p-value (0.1517) > alpha (0.05)

To make sure that all our assumptions were correct and to understand the attributes, we performed hypothesis testing. Click below.





15 November 2021

1

2

3

# Milestone **Two**

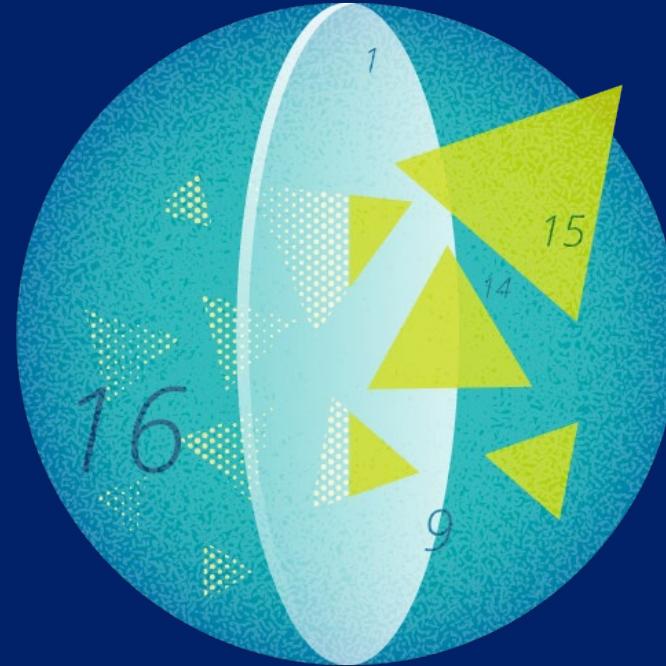
# README.

1

2

3

- 💡 E\_5\_milestone\_2.zip
- ➡️  E\_5\_milestone\_2.pptx
- ➡️  E\_5\_Task\_2.1.html
- ➡️  E\_5\_Task\_2.2.html
- ➡️  E\_5\_Task\_2.3.pbix
- ➡️  E\_5\_Task\_2.4.html



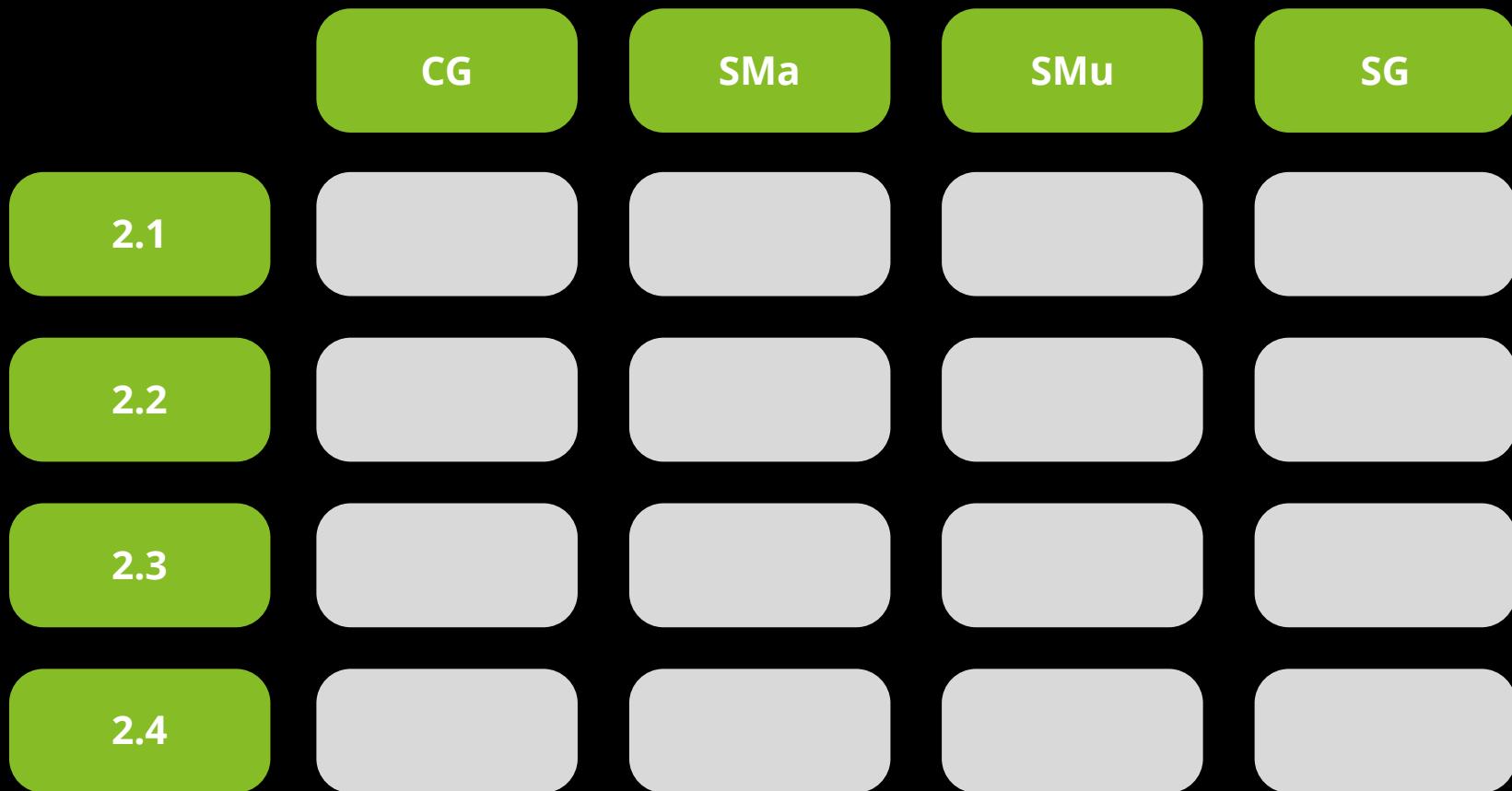
Our final advice to the bank is located at the end of this very PowerPoint presentation

# Individual Contributions.

1

2

3

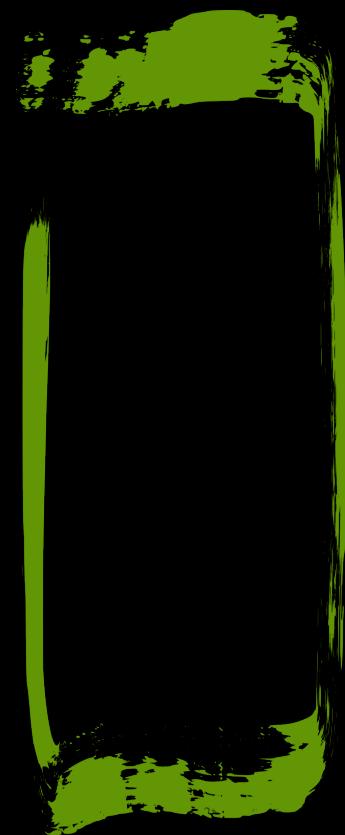


Analyst  
Deck

Presentation by **Chandrasekhar Gudipati**.

# 2.1. Visualization using Python

	CG	SMa	SMu	SG
2.1				
2.2				
2.3				
2.4				



# 2.1. Visualization using Python

1

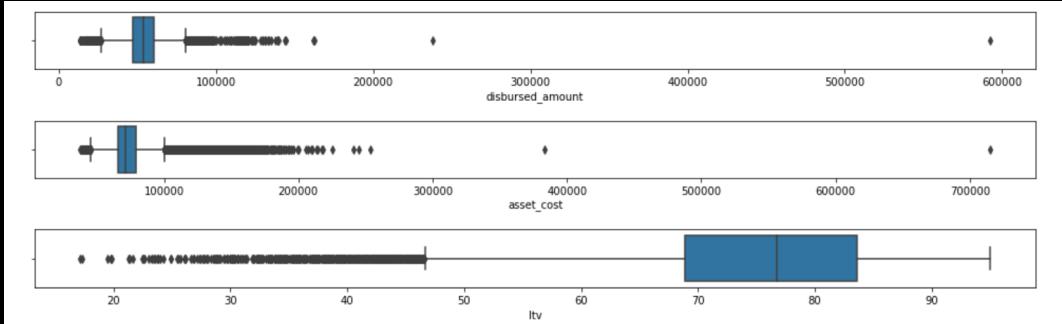
2

3



## To Do List

- Percentage of default
- Imbalanced data
- Missing values
- Outliers
- Analyze default variable with demographic related data
- Association between default and other variables



```
ra = []
comb = {}
for i in range(0, len(ma)):
    ra.append(ma[i] - mi[i])
#print(df1.columns[2])
print("range: \n")
for j in range(0, len(ra)):
    comb[df1.columns[j]] = ra[j]
#print(df1.columns[j], "\t\t\t", ra[j])
#print(comb)
for key, value in comb.items():
    print(key, ' : ', value)

range:

Loan_id : 23314.0
disbursed_amount : 579091.0
asset_cost : 677956.0
ltv : 77.85000000000001
branch_id : 260.0
MobileNo_Avl_Flag : 0.0
Aadhar_flag : 1.0
PAN_flag : 1.0
VoterID_flag : 1.0
Driving_flag : 1.0
Passport_flag : 1.0
PERFORM_CNS.SCORE : 890.0
DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS : 7.0
NO_OF_INQUIRIES : 23.0
loan_default : 1.0
```

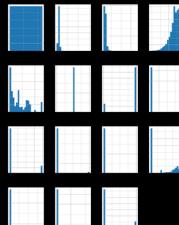
```
df1.quantile(q=0.25)
```

	5829.50
Loan_id	46949.00
disbursed_amount	65629.00
asset_cost	68.83
ltv	13.00
branch_id	1.00
MobileNo_Avl_Flag	1.00
Aadhar_flag	0.00
PAN_flag	0.00
VoterID_flag	0.00
Driving_flag	0.00
Passport_flag	0.00
PERFORM_CNS.SCORE	0.00
DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS	0.00
NO_OF_INQUIRIES	0.00
loan_default	0.00
Name	0.25, dtype: float64

```
df1.var() df1.std()
```

```
df1.skew(skipna = True)
```

```
kurt = df1.kurt()
kurt
```



default: 5126  
total: 23315  
Default%: 21.985%

# 2.1. Visualization using Python

1

2

3



## To Do List



- Percentage of default
- Imbalanced data
- Missing values
- Outliers
- Analyze default variable with demographic related data
- Association between default and other variables



```
total_deleted = total_table.dropna()  
total_deleted
```

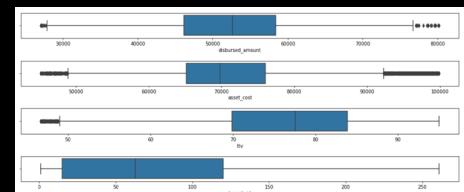
```
print("total_shape rows: ",total_table.shape[0])  
print("total_shape rows: ",total_deleted.shape[0])  
print("lost rows: ",total_table.shape[0]-total_deleted.shape[0])  
print("lost rows percentage: ",((total_table.shape[0]-total_deleted.shape[0])/total_table.shape[0])*100)
```

```
total_shape rows: 23315  
total_shape rows: 22545  
lost rows: 770  
lost rows percentage: 3.3025948959897065
```

Despite my reservations against dropping *n/a* values, we will proceed with **total\_deleted** because that is what the task requires.

```
df2 = df1.dropna()  
  
def Remove_Outlier_Indices(df):  
    Q1 = df.quantile(0.25)  
    Q3 = df.quantile(0.75)  
    IQR = Q3 - Q1  
    trueList = ~((df < (Q1 - 1.5 * IQR)) | (df > (Q3 + 1.5 * IQR)))  
    return trueList  
  
# Index List of Non-Outliers  
nonOutlierList = Remove_Outlier_Indices(df2)  
  
# Non-Outlier Subset of the Given Dataset  
dfSubset = df2[nonOutlierList]
```

```
dfSubset2 = dfSubset.dropna()  
dfSubset2
```

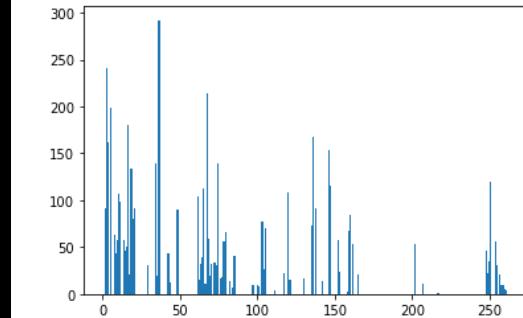


# 2.1. Visualization using Python

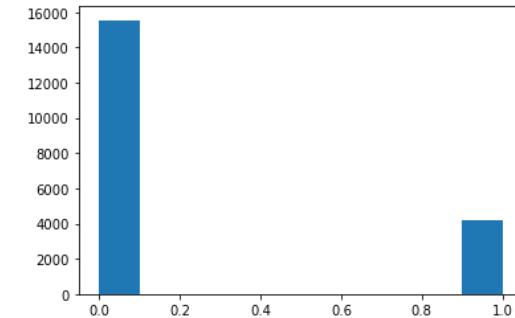
## To Do List

- Percentage of default
- Imbalanced data
- Missing values
- Outliers
- Analyze default variable with demographic related data
- Association between default and other variables

branch\_id (x) and  
count(loan\_default) (y)



loan\_default (x) and  
aadhaar\_flag (y)

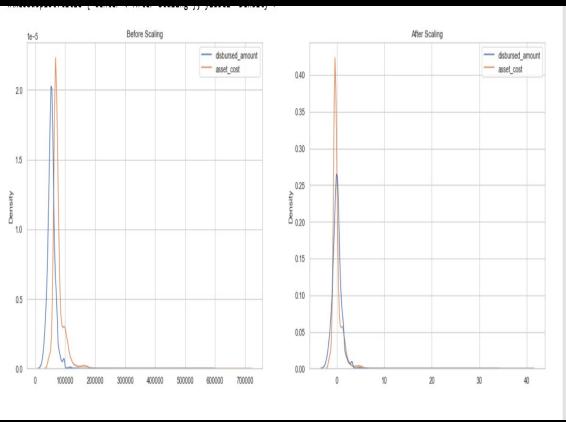
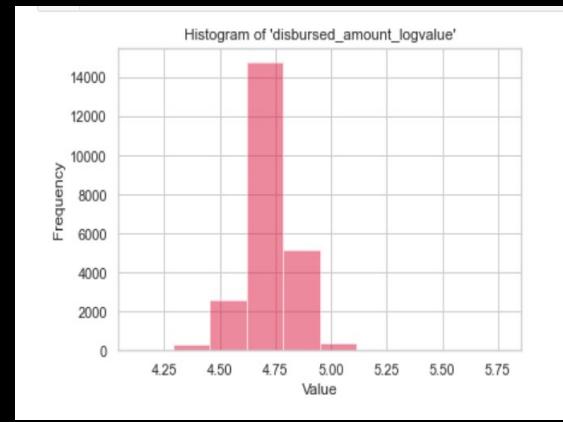
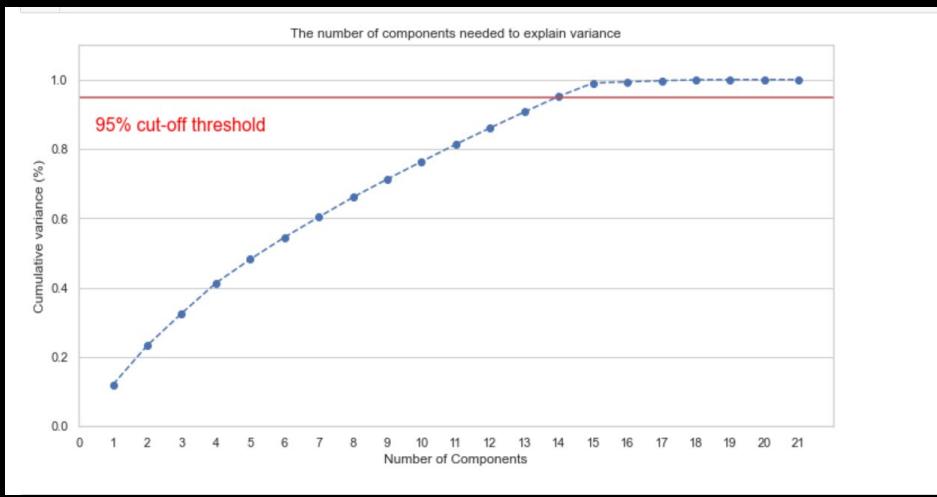
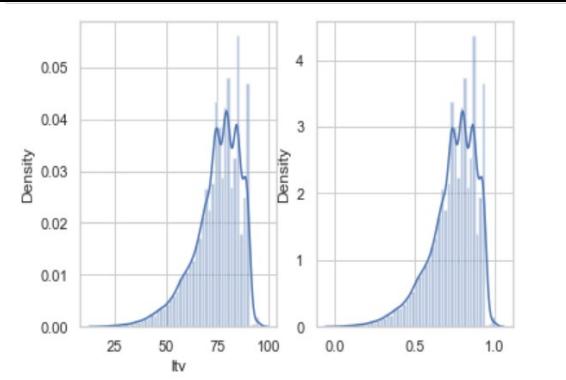
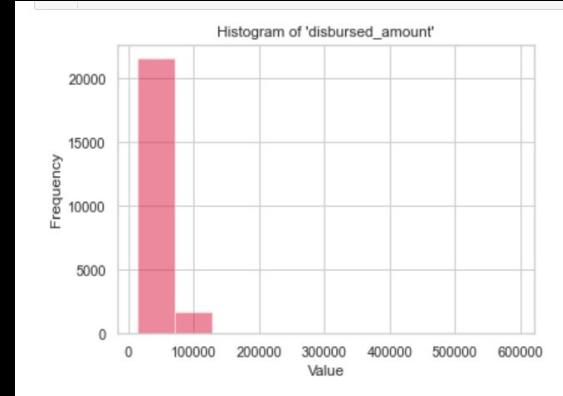
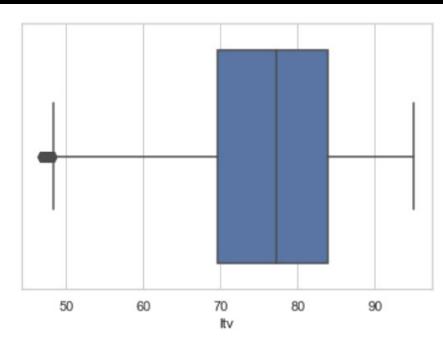
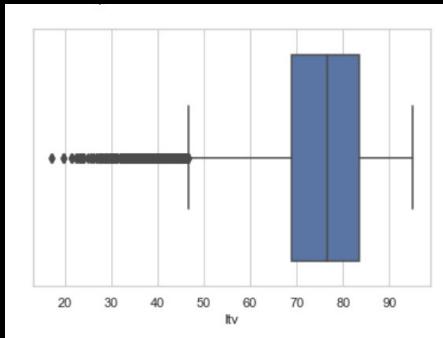


```
total_table.corr()['loan_default']
```

Loan_id	-0.010647
disbursed_amount	0.062805
asset_cost	0.002535
ltv	0.095423
branch_id	0.037361
MobileNo_Avl_Flag	NaN
Aadhar_flag	-0.050624
PAN_flag	0.007771
VoterID_flag	0.046974
Driving_flag	0.004321
Passport_flag	-0.010403
PERFORM_CNS.SCORE	-0.060836
DELINQUENT.ACCTS.IN.LAST.SIX.MONTHS	0.032289
NO.OF_INQUIRIES	0.042042
loan_default	1.000000
Name: loan_default, dtype: float64	



# 2.2. Exploratory Data Analysis

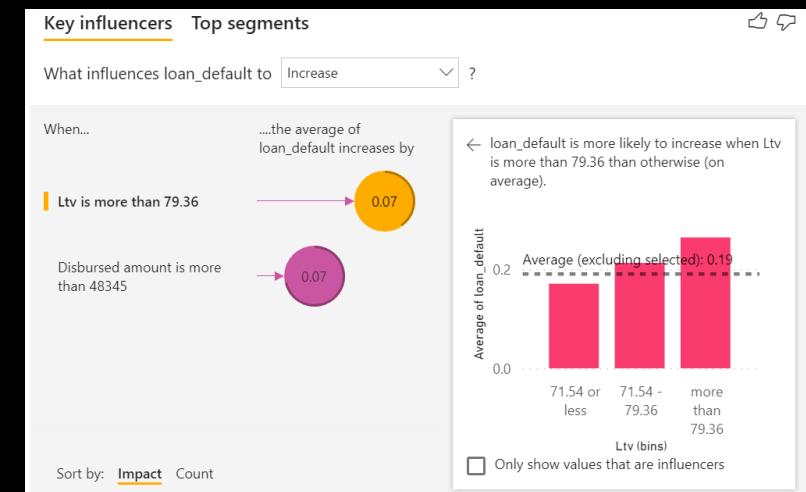
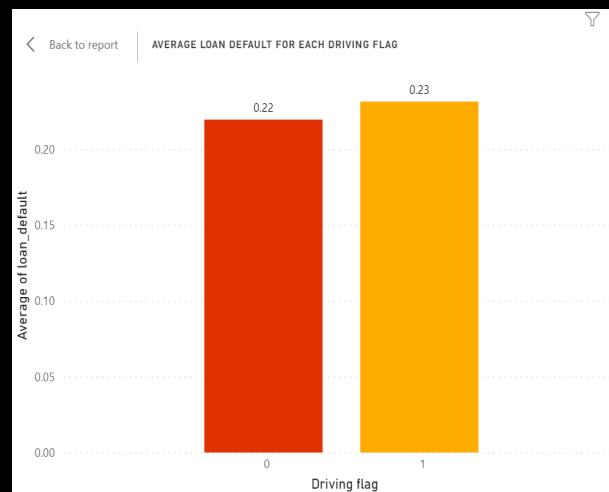
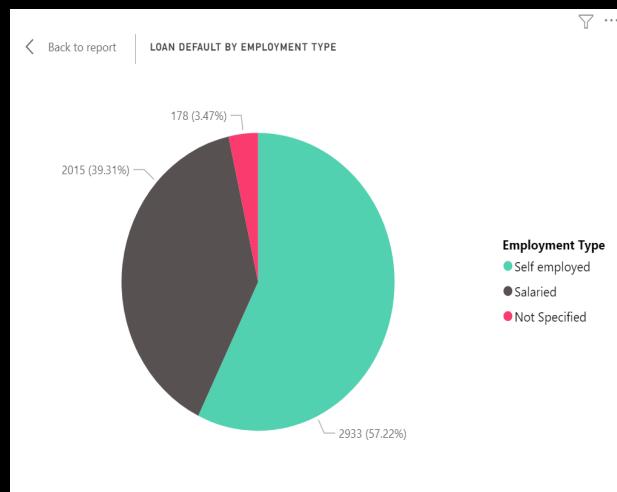


# 2.3. Visualization using Power-BI

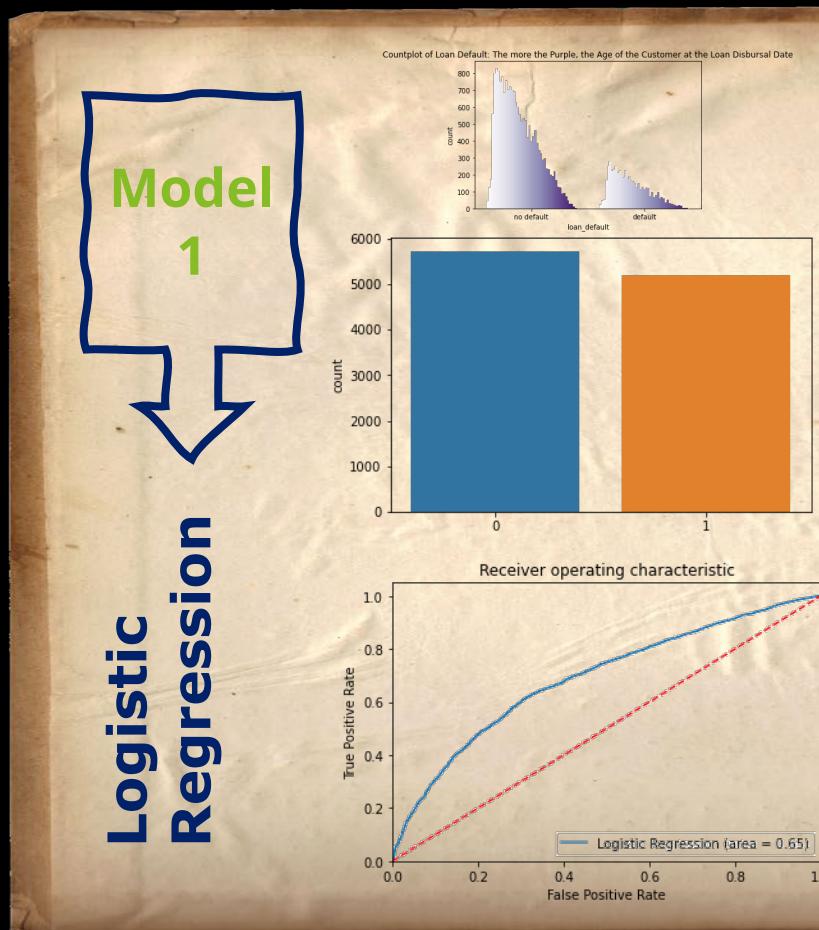
1

2

3

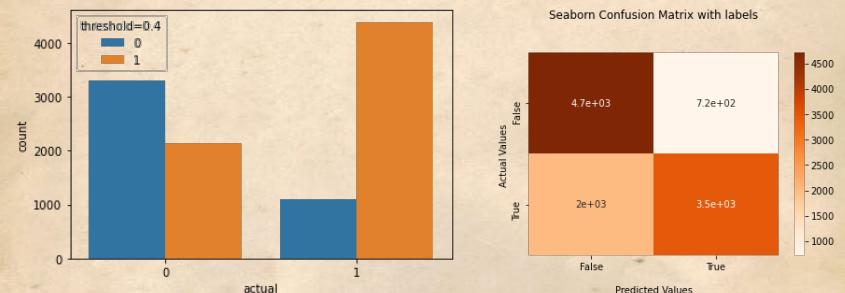


# 2.4. Model Building using ML Algos



All hail Lord SMOTE!

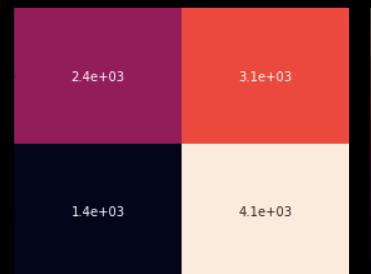
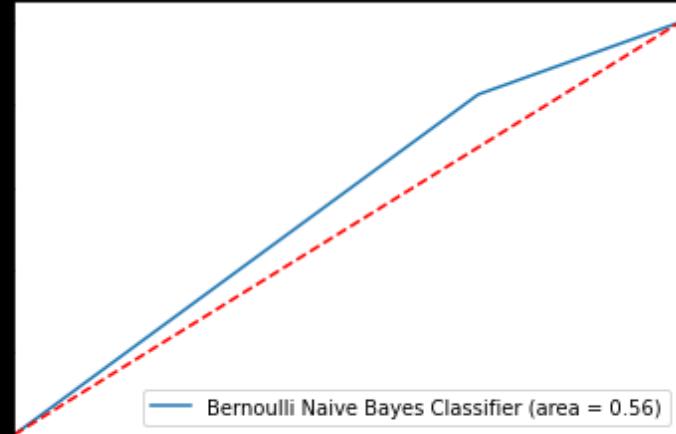
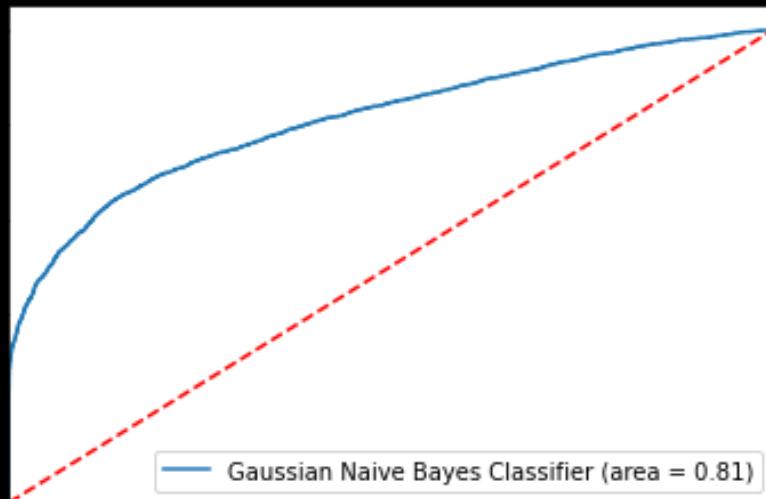
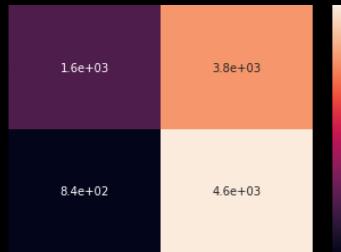
## Model 2 Linear Regression with Manual Thresholds



- \* This 'model' is just an ultra elaborate demonstration of the effects of changing parameters on results and its final results are not to be taken seriously

# 2.4. Model Building using ML Algos

**Model 3:**  
Gaussian  
Naïve Bayes  
Classifier



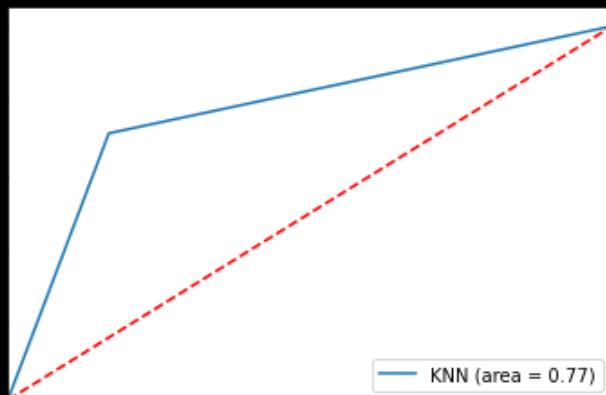
**Model 4:** Bernoulli  
Naïve Bayes Classifier

# 2.4. Model Building using ML Algos



accuracy -> 0.6408

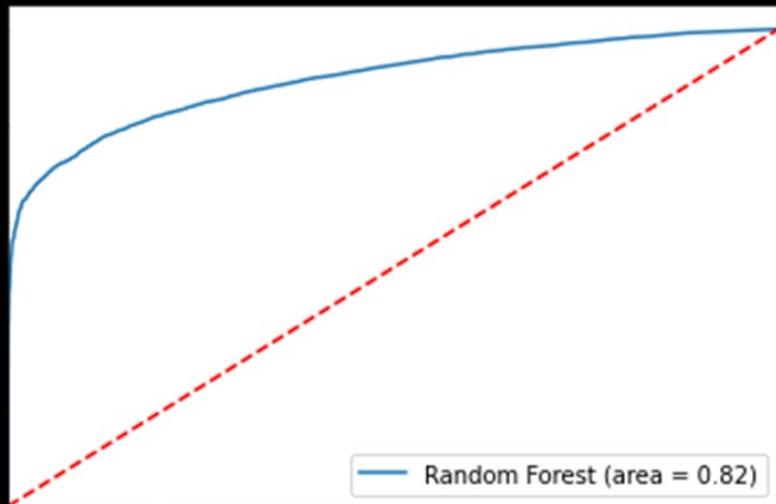
**Model 5:** K Nearest Neighbours



**Model 6:** Decision Trees Classifier

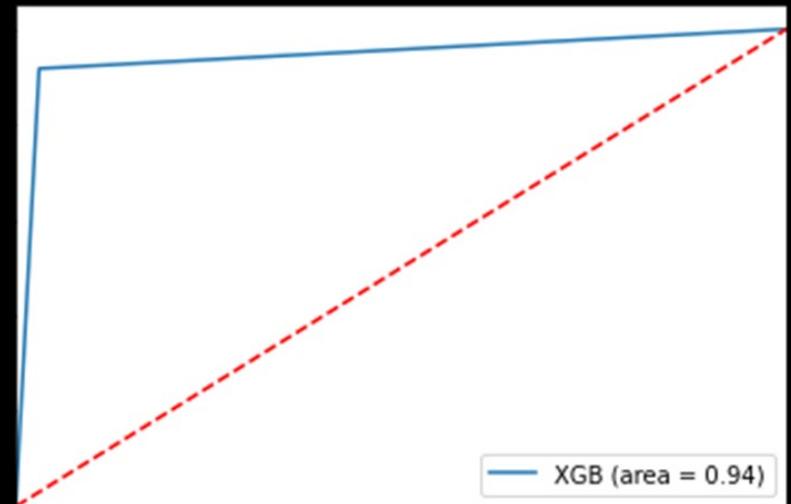
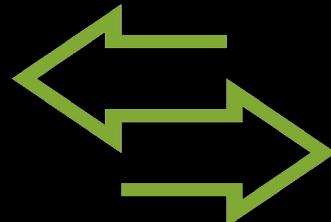
[[4014, 1425],  
[2495, 2980]]

## 2.4. Model Building using ML Algos



accuracy -> 0.8201    [[4834, 605],  
[1358, 4117]]

**Model 7a:** Random Forest

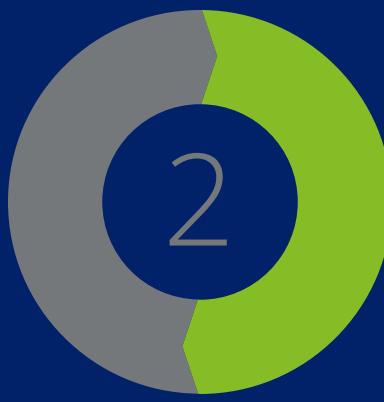


accuracy -> 0.9422    [[5248, 191],  
[ 439, 5036]]

**Model 7b:** XGB

## 2.4. Comparisons

Model	Accuracy	Model	Accuracy
Logistic Regression	0.65	Decision Trees Classifier	0.64
Gaussian	0.81	Random Forest	0.82
Bernoulli	0.56	XGB	0.94
KNN	0.77		



01 December 2021

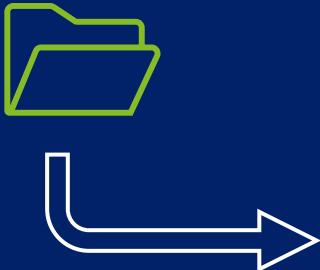
1

2

3

# Milestone Three

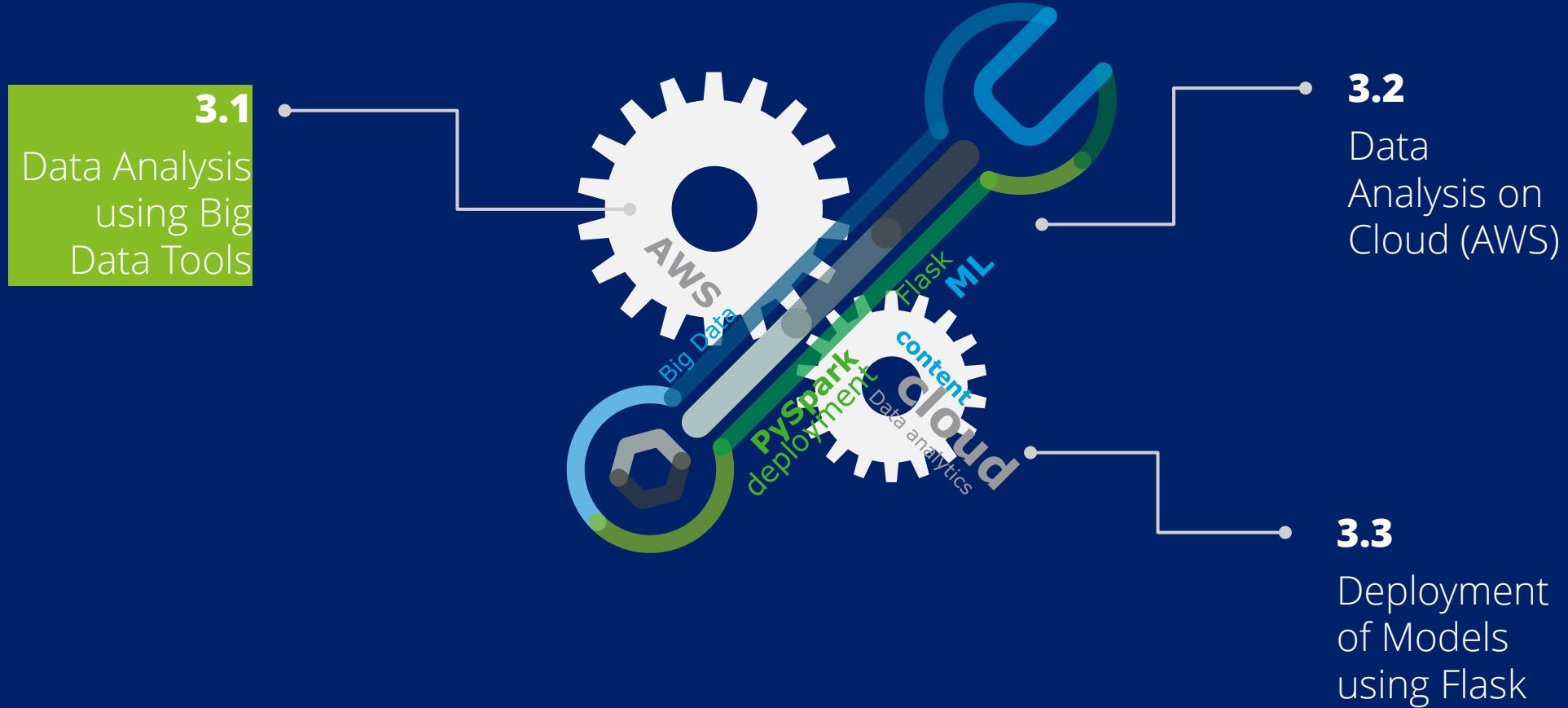
# Read\_me.



1

2

3



# 3.1 Data Analysis using Big Data Tools.

1

2

3

```
df.describe()
```

```
DataFrame[summary: string, disbursed_amount: string, asset_cost: string, ltv: string, PERFORM_CNS_SCORE: string, b  
ranch_id: string, MobileNo_Avl_Flag: string, Aadhar_flag: string, PAN_flag: string, VoterID_flag: string, Driving_  
flag: string, Passport_flag: string, DELINQUENT_ACCTS_IN_LAST_SIX_MONTHS: string, NO_OF_INQUIRIES: string, loan_de  
fault: string, Date_of_Birth_month: string, DisbursalDate_month: string, Date_of_Birth_year: string, DisbursalDate  
_year: string, CREDIT_HISTORY_MONTHS_SCALED: string, age_at_disb_norm: string, Emp_Type: string, region_East: stri  
ng, region_North: string, region_South: string, region_West: string]
```

```
: spark.sql("select DISTINCT region from Data ").show()
```

```
+-----+  
|region|  
+-----+  
| South|  
| East|  
| West|  
| North|  
+-----+
```

```
: spark.sql("select DISTINCT Employment_Type from Data ").show()
```

```
+-----+  
|Employment_Type|  
+-----+  
| Self employed|  
| Salaried|  
+-----+
```

```
defaults_perc_by_region=defaults_and_nonDefaults_by_region.loc[defaults_and_nonDefaults_by_region['loan_default'] =  
= 1]
```

```
defaults_perc_by_region['percentage']=(defaults_perc_by_region['count']/defaults_perc_by_region['count'].sum())*100
```

```
defaults_perc_by_region
```

	region	loan_default	count	percentage
4	West	1	966	19.523040
5	South	1	1132	22.877930
6	North	1	724	14.632175
7	East	1	2126	42.966855

Saving output file in parquet format.

```
Customers_Defaulted.coalesce(1).write.save("ParquetFiles/CustomersDefaulted.parquet", format="parquet")
```

Reading the data from the stored parquet file

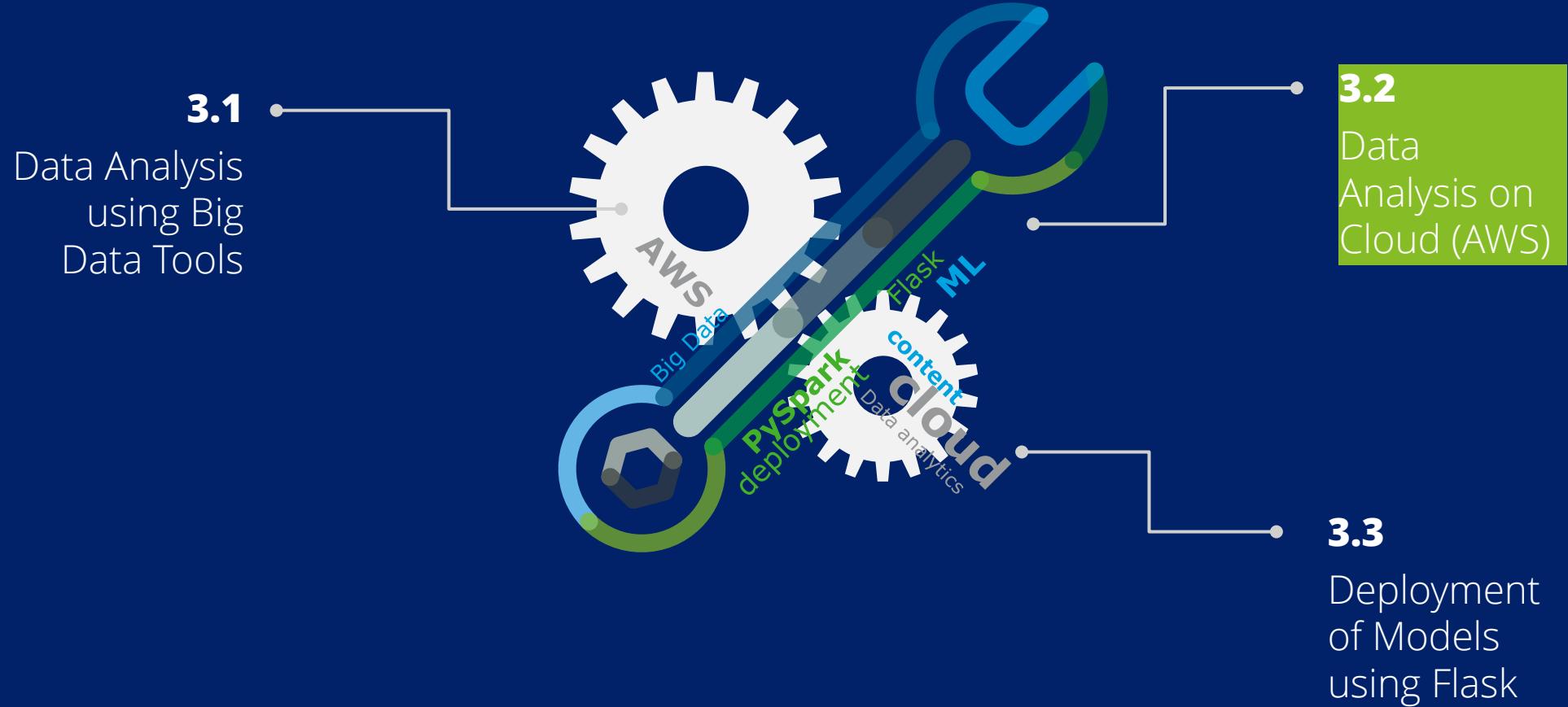
```
parquetFileDF = spark.read.parquet("ParquetFiles/CustomersDefaulted.parquet")
```

```
parquetFileDF.createOrReplaceTempView("DefaultersParquetFile")
```

```
spark.sql("select * from DefaultersParquetFile").show(1)
```

Defaulted_customers_loan_id	disbursed_amount	branch_id
3	55348	2

only showing top 1 row



# Data Analysis on Cloud (AWS).

1  
2  
3

The screenshot shows the AWS S3 console interface. At the top, there are three tabs: 'Redshift', 'redshift-cluster-4 - Redshift quer...', and 'e5capstone - S3 bucket'. The current view is for the 'e5capstone' bucket. The navigation bar includes 'Services' and a search bar. The main area is titled 'e5capstone' and shows the 'Objects' tab selected. Below the tab bar, there are buttons for 'Upload', 'Copy S3 URI', 'Copy URL', 'Download', 'Open', 'Delete', 'Actions', and 'Create folder'. A search bar labeled 'Find objects by prefix' is present. A table lists the object 'total\_table - Copy.csv', which is a CSV file uploaded on November 27, 2021, at 12:05:17 UTC+05:30, with a size of 2.1 MB and a storage class of Standard. The table has columns for checkbox, Name, Type, Last modified, Size, and Storage class. At the bottom, there are links for Feedback, English (US), Privacy, Terms, Cookies, and a copyright notice from 2021. The status bar at the bottom right shows the date as 11/2.

Uploading data to Amazon S3 (Simple Storage Service)



# Data Analysis on Cloud (AWS).

1  
2  
3



Amazon **Redshift**

The screenshot shows the Amazon Redshift Query Editor interface. On the left, a sidebar lists various features: QUERIES (selected), EDITOR (highlighted in orange), DATA SHARES, CONFIG, MARKETPLACE, ADVISOR, ALARMS, and EVENTS. The main area displays a 'Resources' panel with 'Status' set to 'Connected'. It shows two queries: 'Query 1' (green checkmark) and 'Query 2' (orange checkmark). The code editor contains the following SQL query:

```
1 select * from e5_table;
```

Below the code editor are buttons for 'Run', 'Save', 'Schedule', and 'Clear'. A 'Send feedback' link is also present. The bottom of the screen shows a taskbar with icons for File, Home, Recent, Task View, Start, Edge, File Explorer, Microsoft Teams, Google Chrome, and File History. The system tray shows the date (11/2), time (12:1), weather (24°C Rain), and battery status.



Amazon **Redshift**

Uploading data to Redshift (for large structured data, requires predefinition of table)

# Data Analysis on Cloud (AWS).

1

2

3



Amazon **Redshift**

The screenshot shows the AWS Redshift Query Editor v2 interface. On the left sidebar, there are icons for Database, Queries, and Charts. The main area displays a query editor window with the following details:

- Cluster:** redshift-cluster-4 (awsuser)
- Database:** dev
- Query:** COPY dev.public.e5\_table FROM 's3://e5capstone/total\_table - Copy.csv' IAM\_ROLE 'arn:aws:iam::064217613240:role/RedshiftS3ReadWriteAccess' FORMAT AS CSV DELIMITER ',' QUOTE '\"' REGION AS 'us-east-1'
- Result 1:** Summary  
Returned rows: 0  
Elapsed time: 21.8s  
Result set query:  
COPY dev.public.e5\_table FROM 's3://e5capstone/total\_table - Copy.csv' IAM\_ROLE 'arn:aws:iam::064217613240:role/RedshiftS3ReadWriteAccess' --ConsoleRequestID=1-61a1d302-0949f84f163048de2a0cea7c



Amazon **Redshift**

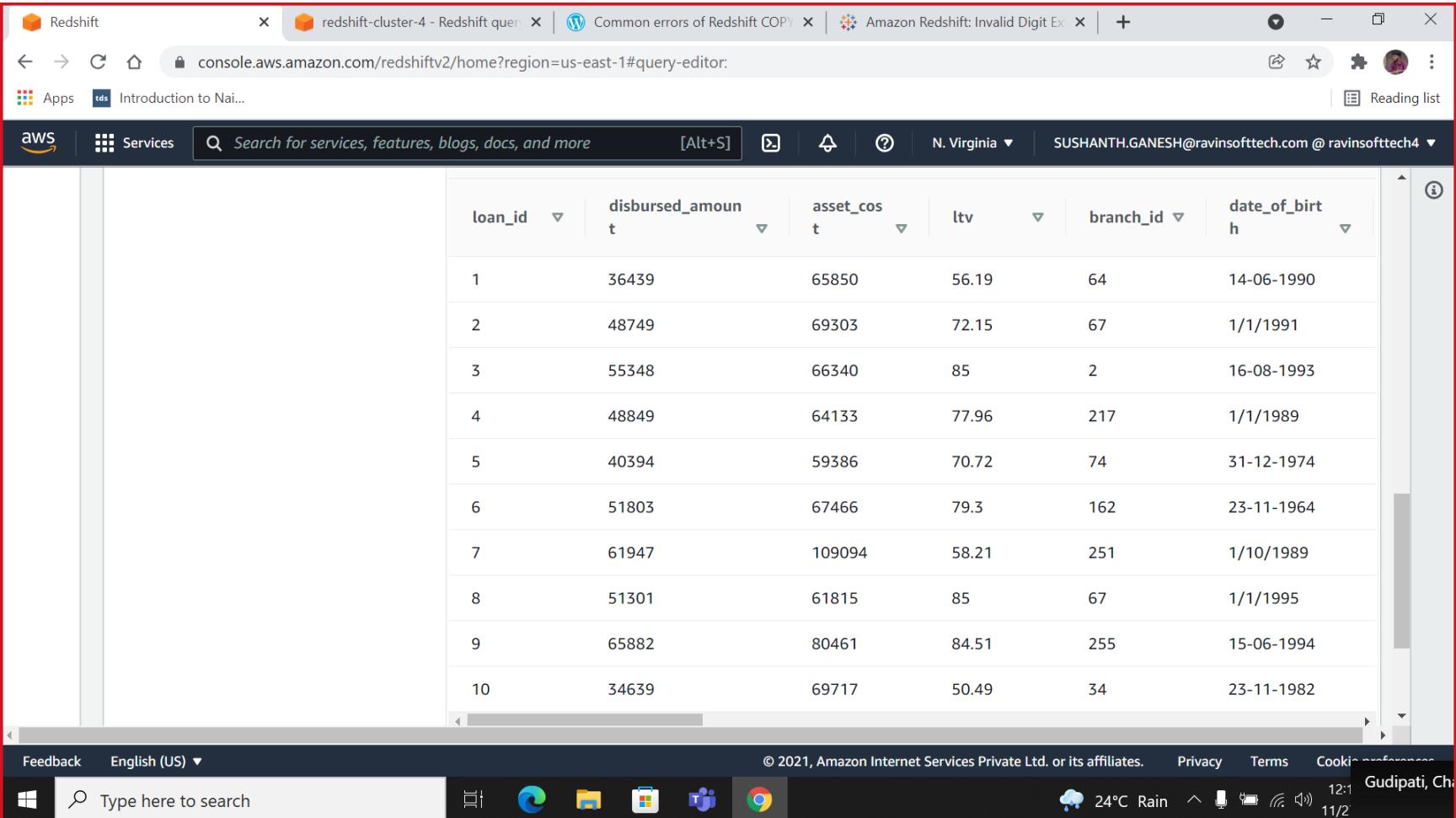
Uploading data to Redshift (for large structured data, requires predefinition of table)

# Data Analysis on Cloud (AWS).

1

2

3



The screenshot shows the AWS Redshift query editor interface. The browser address bar displays the URL: `console.aws.amazon.com/redshiftv2/home?region=us-east-1#query-editor:`. The AWS navigation bar includes the services menu, search bar, and user information: N. Virginia, SUSHANTH.GANESH@ravinsofttech.com @ ravinsofttech4.

The main content area shows a table of loan data with the following columns: loan\_id, disbursed\_amount, asset\_cost, ltv, branch\_id, and date\_of\_birth. The data consists of 10 rows of loan records.

loan_id	disbursed_amount	asset_cost	ltv	branch_id	date_of_birth
1	36439	65850	56.19	64	14-06-1990
2	48749	69303	72.15	67	1/1/1991
3	55348	66340	85	2	16-08-1993
4	48849	64133	77.96	217	1/1/1989
5	40394	59386	70.72	74	31-12-1974
6	51803	67466	79.3	162	23-11-1964
7	61947	109094	58.21	251	1/10/1989
8	51301	61815	85	67	1/1/1995
9	65882	80461	84.51	255	15-06-1994
10	34639	69717	50.49	34	23-11-1982



Amazon **Redshift**



Amazon **Redshift**

Uploading data to Redshift (for large structured data, requires predefinition of table)

# Data Analysis on Cloud (AWS).

1  
2  
3



Amazon Athena



Amazon Athena

The screenshot shows the Amazon Athena Query Editor interface. On the left, there's a sidebar titled 'Data' with sections for 'Data Source' (set to 'AwsDataCatalog') and 'Database' (set to 'db'). Below that is a 'Tables and views' section with a 'Create' button and a search bar. Under 'Tables (2)', there are two entries. The main area is titled 'Query 1 x' and contains the following SQL code:

```
1 CREATE EXTERNAL TABLE IF NOT EXISTS `db`.`e5_athena_table` (
2     `loan_id` int,
3     `disbursed_amount` float,
4     `asset_cost` float,
5     `ltv` float,
6     `branch_id` int,
7     `date_of_birth` string,
8     `employment_type` string,
9     `disbursaldate` string,
10    `mobileno_avl_flag` int,
11    `aadhar_flag` int,
12    `pan_flag` int,
13    `voterid_flag` int,
14    `driving_flag` int,
15    `passport_flag` int,
```

The interface includes tabs for 'Editor', 'Recent queries', 'Saved queries', and 'Settings'. A 'Workgroup' dropdown is set to 'primary'. At the bottom, there are links for 'Feedback', 'English (US)', and copyright information: '© 2021, Amazon Internet Services Private Ltd. or its affiliates.' followed by 'Privacy', 'Terms', and 'Cookie preferences'.

Uploading to Athena (serverless, useful even for unstructured and semi structured data, great for PowerBI integration)

# Data Analysis on Cloud (AWS).

1  
2  
3



Amazon Athena



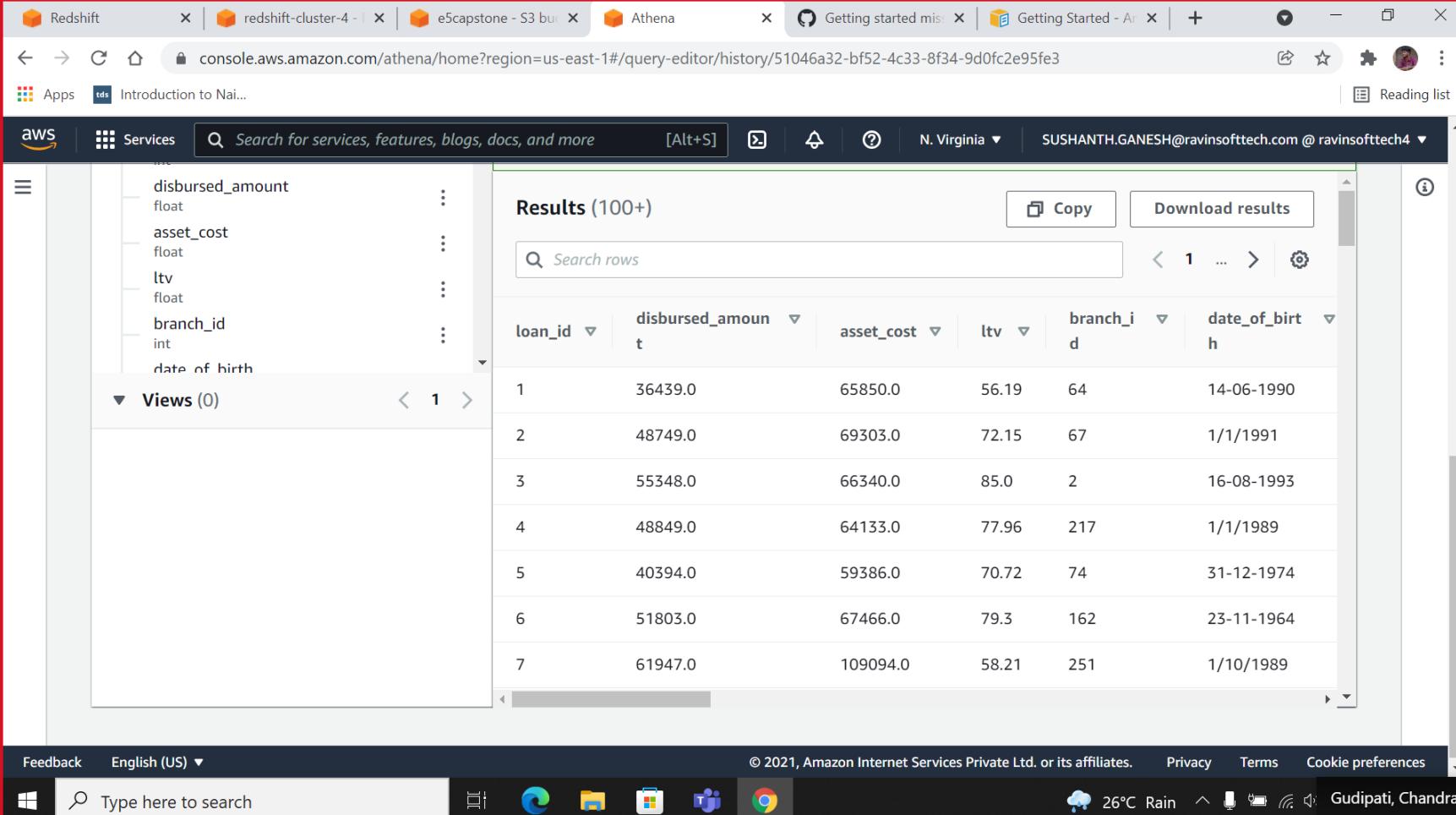
Amazon Athena

The screenshot shows the AWS Athena query editor interface. On the left, there's a sidebar with a search bar and sections for 'Tables (2)' and 'Views (0)'. Under 'Tables', two tables are listed: 'carsalesdata' and 'e5\_athena\_table'. The 'e5\_athena\_table' table has several columns: loan\_id (int), disbursed\_amount (float), asset\_cost (float), ltv (float), branch\_id (int), and date\_of\_birth. On the right, the main area displays a SQL query with several lines of code. Below the code, there are buttons for 'Run again', 'Cancel', 'Save as', 'Clear', and 'Create'. A success message box is open, stating 'Completed' with a green checkmark, and providing metrics: Time in queue: 0.097 sec, Run time: 0.426 sec, and Data scanned: -. At the bottom of the screen, there's a taskbar with icons for Feedback, English (US), a search bar, and system status indicators like weather, temperature, and location.

Uploading to Athena (serverless, useful even for unstructured and semi structured data, great for PowerBI integration)

# Data Analysis on Cloud (AWS).

1  
2  
3



The screenshot shows the Amazon Athena console interface within a web browser. The left sidebar displays the schema of a table with columns like `disbursed_amount`, `asset_cost`, `ltv`, `branch_id`, and `date_of_birth`. The main area shows a preview of 100+ results from a query, including columns `loan_id`, `disbursed_amount`, `asset_cost`, `ltv`, `branch_id`, and `date_of_birth`. The data includes various numerical values and dates. The browser's address bar shows the URL `console.aws.amazon.com/athena/home?region=us-east-1#/query-editor/history/51046a32-bf52-4c33-8f34-9d0fc2e95fe3`.

loan_id	disbursed_amount	asset_cost	ltv	branch_id	date_of_birth
1	36439.0	65850.0	56.19	64	14-06-1990
2	48749.0	69303.0	72.15	67	1/1/1991
3	55348.0	66340.0	85.0	2	16-08-1993
4	48849.0	64133.0	77.96	217	1/1/1989
5	40394.0	59386.0	70.72	74	31-12-1974
6	51803.0	67466.0	79.3	162	23-11-1964
7	61947.0	109094.0	58.21	251	1/10/1989



Amazon Athena



Amazon Athena

Uploading to Athena (serverless, useful even for unstructured and semi structured data, great for PowerBI integration)

# Data Analysis on Cloud (AWS).

1  
2  
3



Datalake

The screenshot shows the AWS Lake Formation console interface. The left sidebar menu includes options like Dashboard, Data catalog (with Databases selected), Tables, Data filters, Settings, Register and ingest (with Data lake locations and Blueprints), Crawlers, Jobs, and Permissions (with Administrative roles and tasks and LF-Tags). The main content area displays the details for the database 'lake-sush'. The database name is 'lake-sush', the Amazon S3 path is 's3://lake-buck-sush', and there is a checked checkbox for 'Use only IAM access control for new tables in this database'. Below this, the 'LF-Tags (0)' section is shown with a search bar and a button to 'Edit LF-tags'. The browser's address bar shows the URL 'console.aws.amazon.com/lakeformation/home?region=us-east-1#database-details/lake-sush?catalogId=064217613240'.



Datalake

Datalake is a centralized repository ready for large amounts of both structured and unstructured data

1

2

3

# Data Analysis on Cloud (AWS).

Power BI



The screenshot shows the Power BI Data Navigator interface. The left pane displays a tree view of data sources, with 'Amazon Redshift [2]' expanded, showing 'catalog\_history' and 'public' as child nodes. The main pane is titled 'Navigator' and contains the message 'No items selected for preview'. The bottom pane shows a detailed list of fields from the 'e5\_table' in the 'catalog\_history' database, including various summary measures like  $\sum aadhar\_flag$ ,  $\sum asset\_cost$ , etc.

No items selected for preview

Fields

- e5\_table
  - $\sum aadhar\_flag$
  - $\sum asset\_cost$
  - $\sum branch\_id$
  - $\sum credit\_history\_l...$
  - $\sum date\_of\_birth$
  - $\sum delinquent\_acc...$
  - $\sum disbursaldate$
  - $\sum disbursed\_amo...$
  - $\sum driving\_flag$
  - $\sum employment\_ty...$
  - $\sum loan\_default$
  - $\sum loan\_id$
  - $\sum ltv$
  - $\sum mobileno\_avl\_fl...$
  - $\sum no\_of\_inquiries$
  - $\sum pan\_flag$

Uploading to PowerBI

Power BI



# Data Analysis on Cloud (AWS).

1

2

3

The screenshot shows the Power BI Data Navigator interface. On the left, the 'Navigator' pane displays a tree view of data sources. Under 'Amazon Redshift [2]', the 'public [9]' schema is selected, revealing tables like category, date, e5\_table, event, last\_table, listing, sales, users, and venue. The main workspace is titled 'No items selected for preview'. To the right, the 'Fields' pane lists all columns from the selected 'e5\_table', including aadhar\_flag, asset\_cost, branch\_id, credit\_history\_l..., date\_of\_birth, delinquent\_acc..., disbursaldate, disbursed\_am..., driving\_flag, employment\_ty..., loan\_default, loan\_id, ltv, mobileno\_avl\_fl..., no\_of\_inquiries, and pan\_flag. At the bottom, a taskbar includes 'Load', 'Transform Data', and 'Cancel' buttons, along with system status icons.



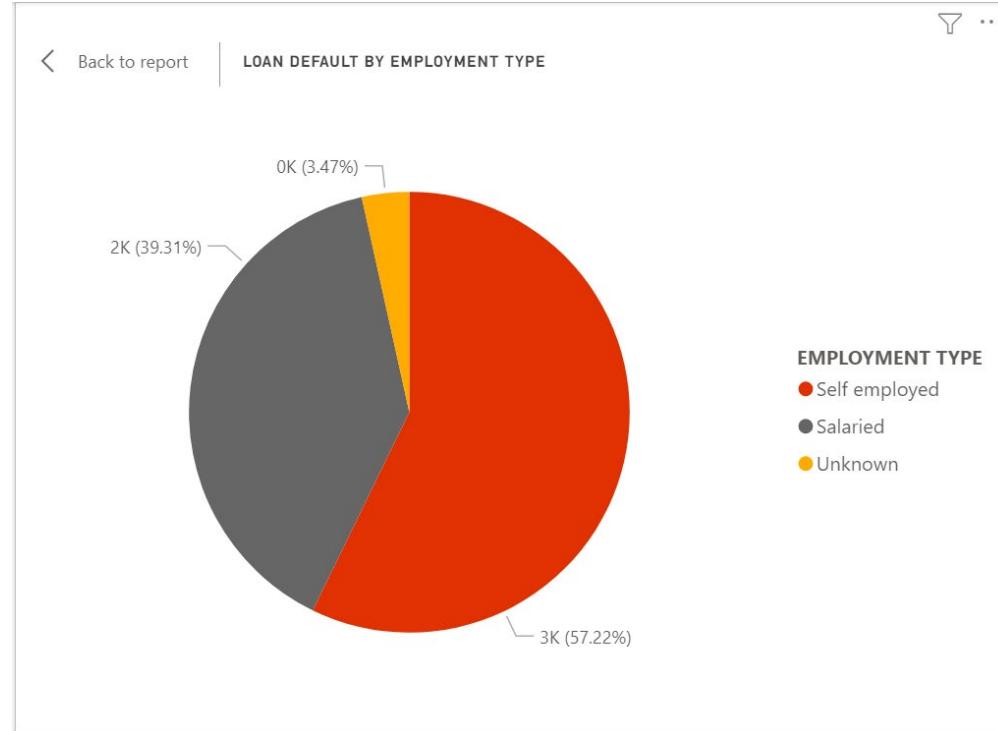
Uploading to PowerBI

1

2

3

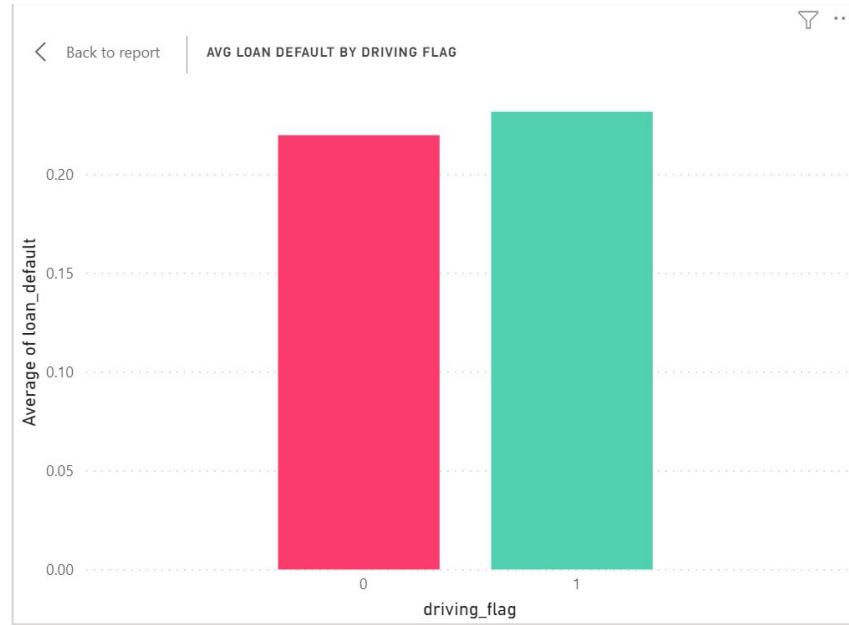
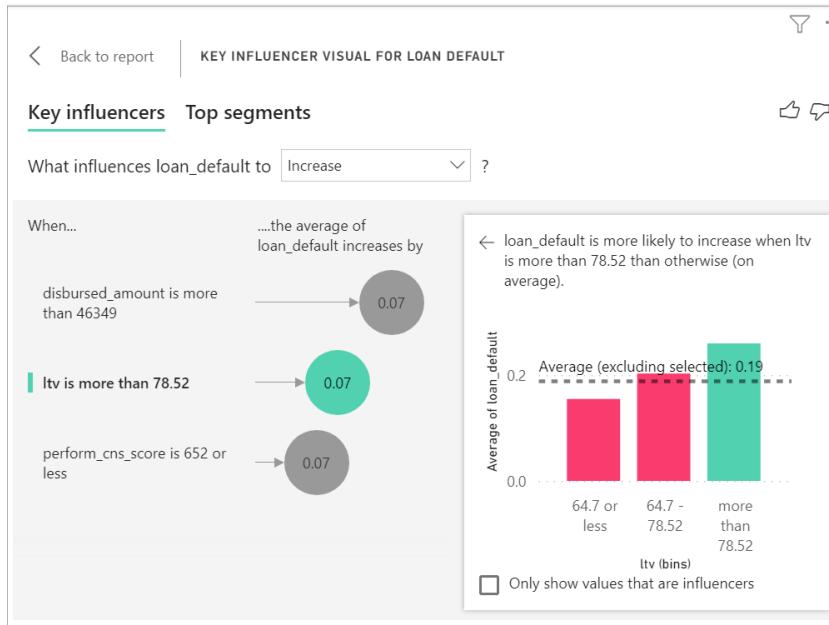
# Data Analysis on Cloud (AWS).



Uploading to PowerBI

# Data Analysis on Cloud (AWS).

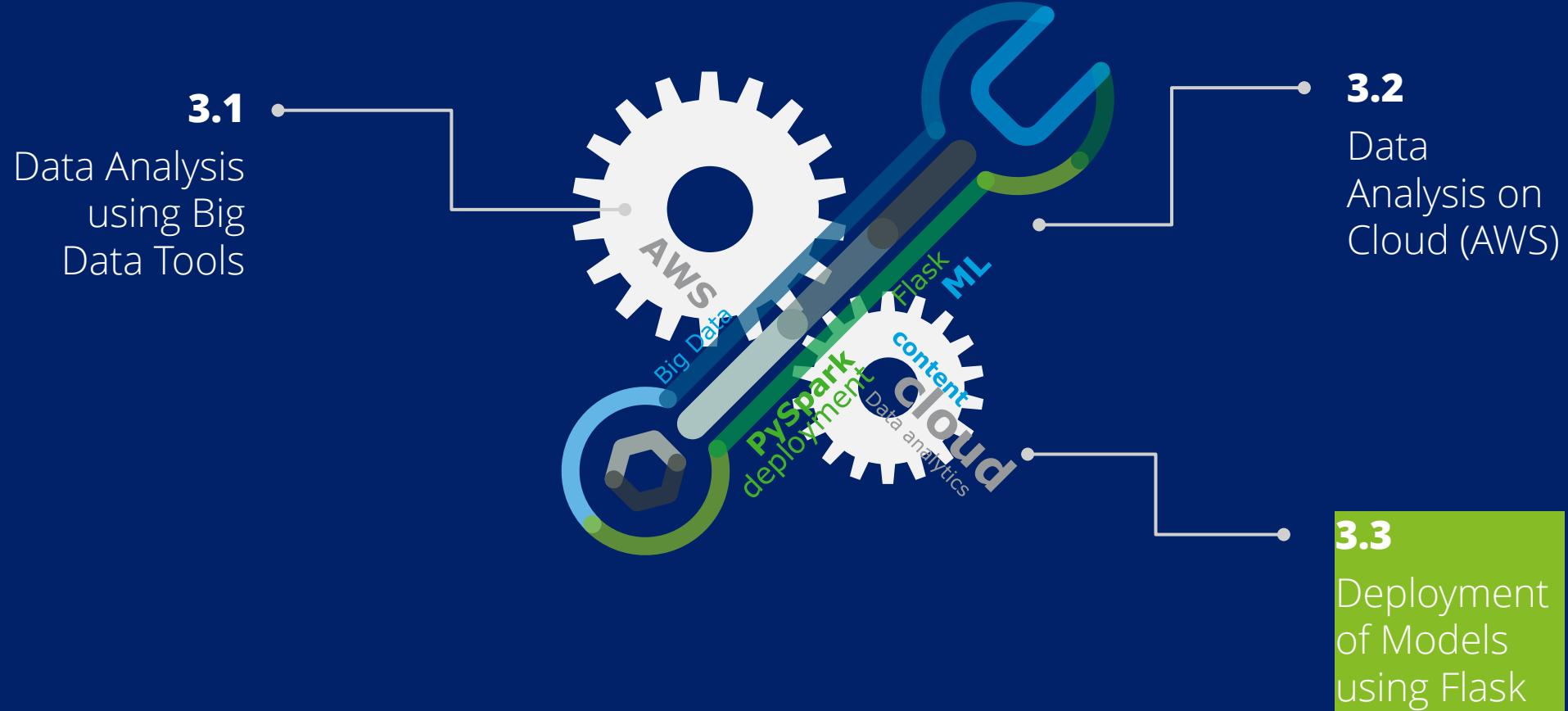
Power BI



Power BI



Uploading to PowerBI



# Deployment using Flask.

1

2

3



Flask

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the "DEPLOYMENT" folder: .ipynb\_checkpoints, templates (index.html, results.html, styles.css), and app.py.
- Code Editor:** The active file is app.py, containing Python code for a Flask application. The code imports numpy, pandas, Flask, request, jsonify, render\_template, and pickle. It defines two routes: a home route that returns a rendered template, and a predict route that takes POST requests, processes form data, creates a DataFrame, makes a prediction, and returns a rendered template with the result.
- Terminal:** Shows log messages indicating a detected change in the app.py file, restarting with windowsapi reloader, and starting a debugger on port 127.0.0.1:5000.
- Status Bar:** Shows Python 3.8.8 64-bit ('base': conda) and other status indicators.



Flask

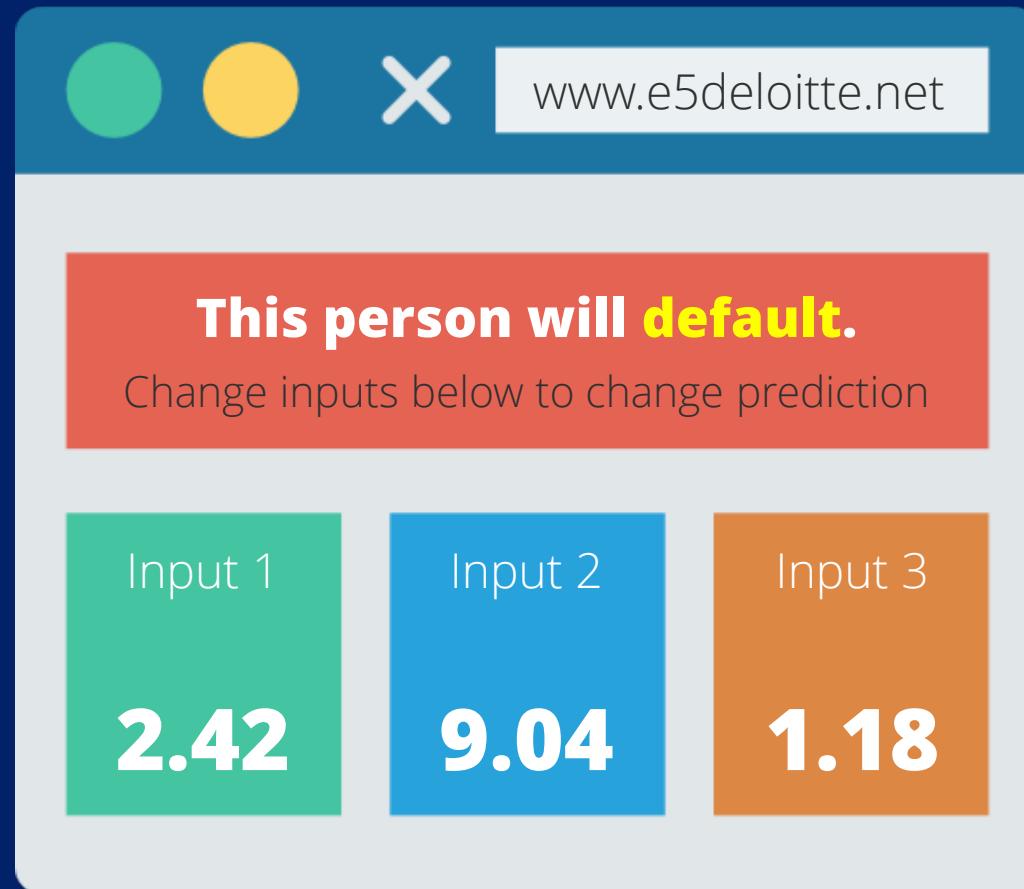
# Deployment using Flask.

A screenshot of a web browser window titled "Deloitte" with the URL "127.0.0.1:5000/predict". The page has a dark blue background and features a "Deloitte E5 Prediction Model" header. Below the header are six input fields: "disbursed\_amount", "asset\_cost", "ltv", "CNS\_Score", "branch\_id", and "age\_at\_disb\_norm". There is also a dropdown menu labeled "Employment-Type". A green "Predict" button is centered below the inputs. The text "Not a defaulter, go ahead with the loan!" is displayed in green. At the bottom, there is a table with the following data:

disbursed_amount	0.039838298
asset_cost	0.042215129
ltv	0.501734104
PERFORM_CNS_SCORE	737
branch_id	64
age_at_disb_norm	0.217391304
Employment_Type_Selfemployed	1



# Problem Statement.



Pretty close, huh?

## **Our recommendations as Deloitte Analysts to the Bank**

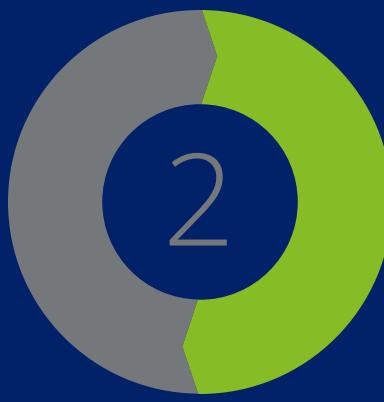
- 

Always check your customer's loan to value ratio. It seems to always be the biggest player in predicting whether your customer will default.  
The more the ltv, the more likely the default (check task 2.4).

Use the XGB Model that we are leaving you with (check task 2.4) for loan default predictions. It gives the best results of all our models.

Prefer salaried employees over self-employed people as customers. They are more likely to have a stable income and hence less likely to default (check task 2.3).





The End.

# Know us better.



Chandrasekhar  
Gudipati



Shruti  
Maigur



Sushant  
Mudalgi



Sushanth  
Ganesh



# Any Questions?



# Deloitte.

## Thank You

