



ROOT CAUSE PREDICTION

BY

SUPERVISED LEARNING

Interim VP, Network Management and Operations:

ALAN ASSELSTINE

PREPARED BY:

SARA KHOSRAVI

CONTENTS

- 1 Project Overview
- 2 Hypothesis
- 3 Data Cleaning
- 4 EDA
- 5 Feature Engineering
- 6 Model Generation
- 7 Conclusion and Recommendation



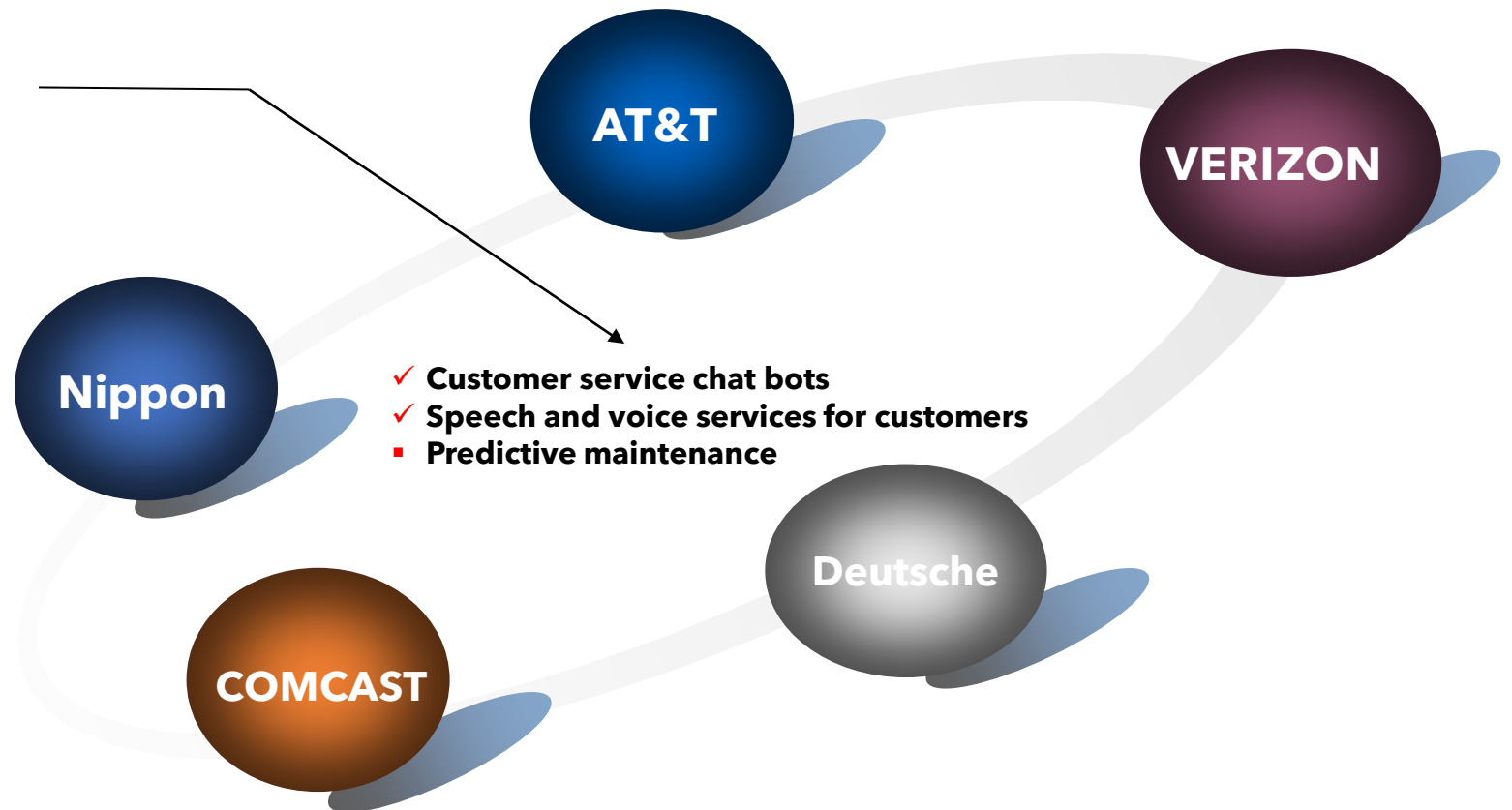
Project Overview

INTELLIGENCE APPLICATIONS

TOP NETWORK COMPANY

Telecommunications is one of the **fastest-growing industries** as well as one that uses artificial intelligence and machine learning in many aspects of their business from enhancing the customer experience to predictive maintenance to improving network reliability.

Predictive maintenance - The ability to fix problems with telecom hardware (such as cell towers, power lines, etc) before they happen, by detecting signals that usually lead to failure



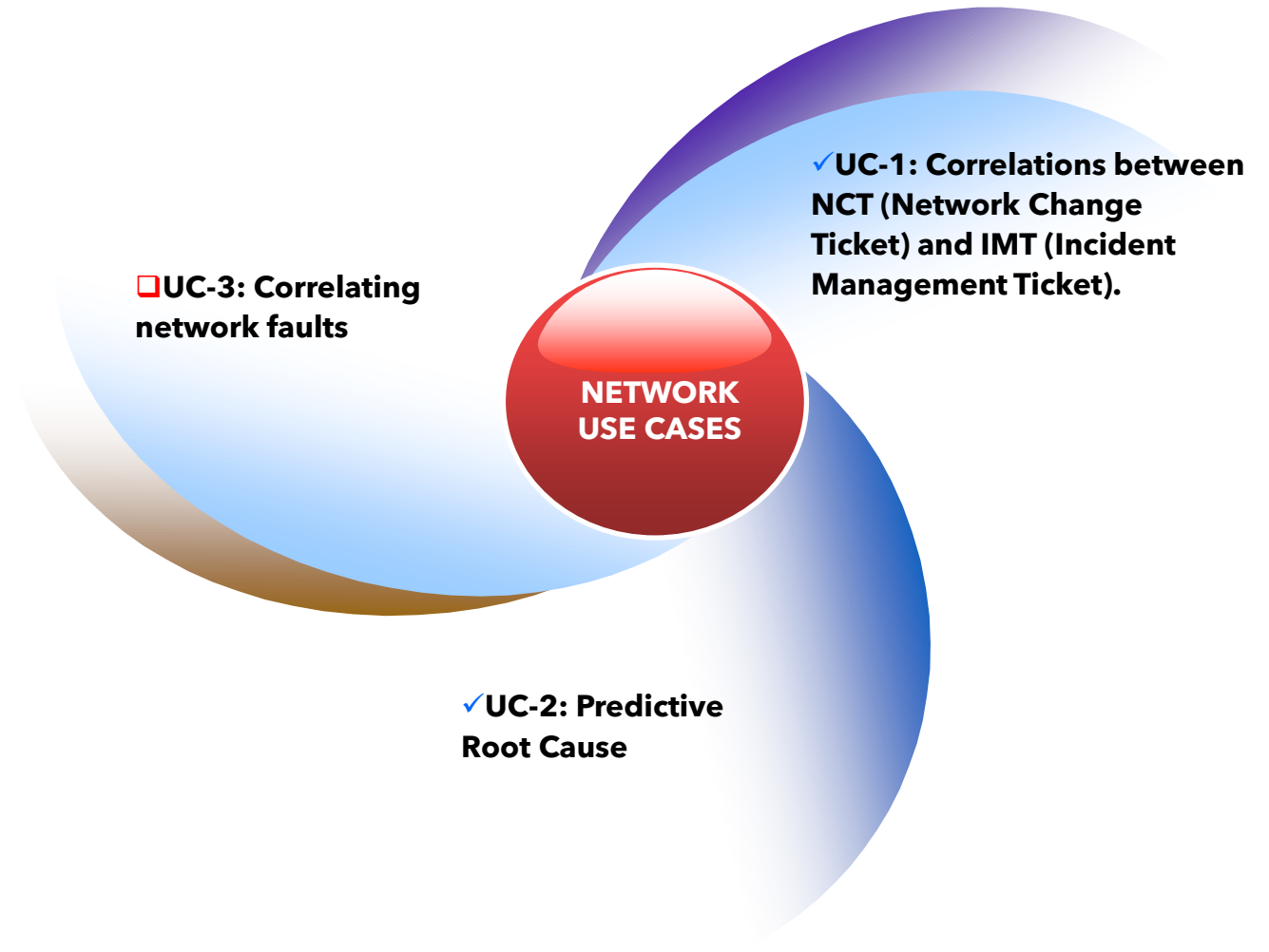
INTRODUCE **USE CASES** AI/ML

REVIEW

ACCORDING TO THE AI NERVE CENTRE ,
THEY HAVE ALREADY STARTED 10
PROJECTS.

SOURCE: Community of Practice - AI Nerve
Centre: ANC – AI Nerve Centre community
of practice JULY 2022

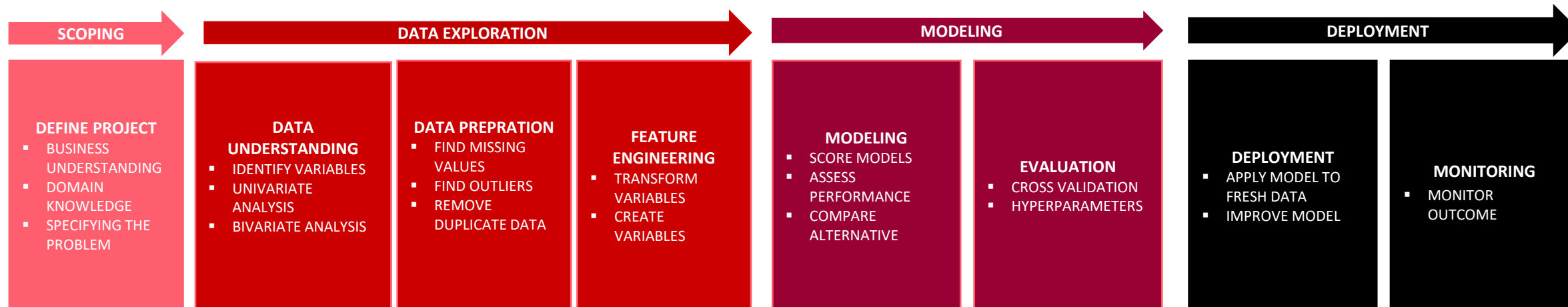
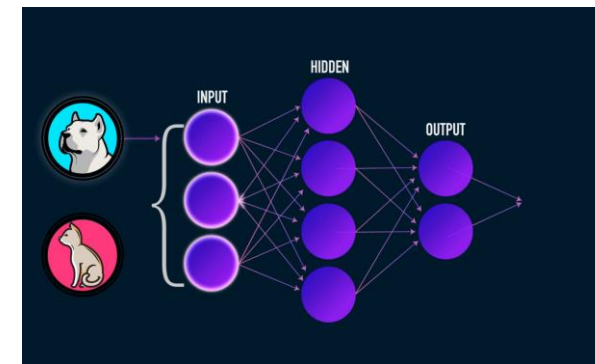
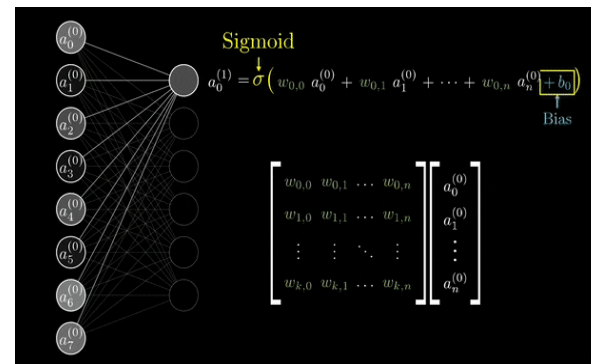
FURTHERMORE, TEAMS HAVE THEIR OWN
ML/AI PROJECTS, LIKE OUR GROUP.



INTRODUCE ML/AI

REVIEW

Machine learning algorithms use **computational** and **statistical** methods to directly use data instead of using predicted values that might act like a model. The efficiency of machine learning algorithms improves as the number of samples and data increases during the process. **Acceleration time** from analytics can help facilitate capacity planning for new services.



THE PROBLEM

REVIEW UC2 PREDICTIVE ROOT CAUSE

- ❑ Rogers Company has collected a data for network and would like to analyze the dataset for Root Cause Prediction.
- ❑ This report consists of a preliminary exploratory data analysis and a classification model among independent variable and target using Python and provide a best fit.
- ❑ The **data contains** Network Automation attributes from **November 2021 to YTD**.
- ❑ The **source** of data is provided by **Remedy** and **ESAP**.
- ❑ The **aim** is to build a **predictive model** and find out the **Root Cause** of the Network.
- ❑ Using this model will try to understand the Network and predict the root cause.
- ❑ Using **supervised learning model** will try to understand the climate and predict the temperature. The models build in the project are as follows:

1. Logistic Regression

2. SVC Model

3. K-Nearest Neighbors

4. Decision Tree Model

5. Random Forest Model

5. XGBoost Model

6. AdaBoost Model



THE ENVIRONMENT

INSTALLING PACKAGES AND LIBRARIES

USING SKLEARN FOR CLASSIFICATION



```
#import libraries for modeling
from sklearn.utils import shuffle
from sklearn.preprocessing import StandardScaler, MinMaxScaler, Normalizer
from sklearn.model_selection import train_test_split
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import cross_val_score
from sklearn import metrics
import math
from sklearn.decomposition import PCA
from sklearn.model_selection import KFold
from sklearn.tree import plot_tree
from sklearn.linear_model import LassoCV
from sklearn.linear_model import RidgeCV
from sklearn.pipeline import Pipeline
from sklearn.svm import SVC
from sklearn.svm import LinearSVC
from sklearn import tree
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.experimental import enable_hist_gradient_boosting # noqa
from sklearn.linear_model import LogisticRegressionCV
from sklearn.linear_model import SGDClassifier
from sklearn.ensemble import VotingRegressor
!pip install xgboost
from xgboost import XGBClassifier
import xgboost as xgb
from sklearn.ensemble import HistGradientBoostingRegressor
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn import metrics
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score, confusion_matrix, roc_curve, auc, roc_auc_score
from sklearn.metrics import classification_report
```


THE ENVIRONMENT

READING THE DATABASE

SETTING UP WORKING DIRECTORY

```
In [1]: # SETTING UP WORKING DIRECTORY
import os
os.getcwd()
os.chdir(r'C:\Users\iramk\OneDrive\Desktop\Data Science Class Note\MLBDA\MLProject')
os.getcwd()

Out[1]: 'C:\\Users\\iramk\\OneDrive\\Desktop\\Data Science Class Note\\MLBDA\\MLProject'
```

Train Dataset: 229986 records and **19** features
Test Dataset: 6283 records and **18** features
ESAP Dataset: 6762 records and **8** features

```
In [4]: #Read files:

# READING THE DATA

#Row data for Train
df = pd.read_excel(r'C:\Users\sara.khosravi\Documents\Sara\Machine Learning\Data\Raw_Data_YTD-2022-08-08.xlsx',na_values=missi
print(df.shape)

#ESAP
dfesap = pd.read_excel(r'C:\Users\sara.khosravi\Documents\Sara\Machine Learning\Data\ESAP.xlsx',na_values=missing_value_format
print(dfesap.shape)

#Test Data for Current Month
dftest = pd.read_excel(r'C:\Users\sara.khosravi\Documents\Sara\Data Analysis\ML\Data\Raw_Data__2022-04-04.xlsx',na_values=mis
print(dfest.shape)

(229986, 19)
(6762, 8)
(6283, 18)
```



HYPOTHESIS

THE HYPOTHESIS

POSSIBLE OUTCOME

- ❑ FOR IMT TICKETS RESOLVED BY NOC, THERE ARE ROOT CAUSE AND ACTION TAKEN THAT RESOLVE EACH ISSUE. THE GOAL IS TO PRE-DETERMINE AN APPROPRIATE ACTION/ROOT CAUSE FOR FUTURE IMT BASED ON HISTORICAL INFORMATION.
- ❑ We are interested in knowing if there is a relationship between 'ROOTCAUSE' and ('rule name', 'Resolution', 'Responsibility', 'Bot', 'month',...).

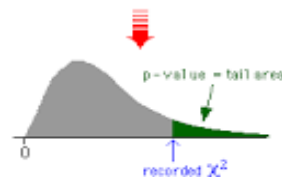
Summary statistic
(helps distinguish H_0 and H_A)

Test statistic
(standard distribution with no unknown parameters under H_0)

P-value
(probability of more 'extreme' test statistic)

$$\chi^2 = \sum \frac{(y_{xy} - e_{xy})^2}{e_{xy}}$$

$$\chi^2 \sim \text{chi-squared } ((r-1)(c-1) \text{ df})$$



```
In [65]: from scipy.stats import chi2_contingency
def chi_square(c1,c2):
    chi_2, p_val, dof, exp_val = chi2_contingency(pd.crosstab(df[c1],df[c2],margins = False))# make sure margins = False

    print(exp_val)
    #print('\nChi-square is : %f'%chi_2, '\nnp_value is : %f'%p_val, '\nndegree of freedom is : %i'%dof)
    print(f'\nChi-square is : {chi_2}', f'\nnp_value is : {p_val}', f'\nndegree of freedom is : {dof}')

    if p_val < 0.05:# consider significant level is 5%
        print("\nThere is some correlation between the two variables at 0.05 significant level")
    else:
        print("\nThere is no correlation between the two variables")
```

```
In [67]: chi_square("RootCause", 'Resolution')

[[7.34294542e-02 1.75592173e-02 7.50576725e+00 ... 5.42739444e-02
 4.78887745e-03 3.03295572e-02]
 [7.56702369e+00 1.80950566e+00 7.73481421e+02 ... 5.59301751e+00
 4.93501545e-01 3.12550978e+00]
 [3.52935118e-01 8.43975283e-02 3.60761071e+01 ... 2.60865088e-01
 2.30175077e-02 1.45777549e-01]
 ...
 [2.07307930e+00 4.95736354e-01 2.11904758e+02 ... 1.53227600e+00
 1.35200824e-01 8.56271885e-01]
 [4.73738414e-04 1.13285273e-04 4.84243048e-02 ... 3.50154480e-04
 3.08959835e-05 1.95674562e-04]
 [1.32646756e-02 3.17198764e-03 1.35588054e+00 ... 9.80432544e-03
 8.65087539e-04 5.47888774e-03]]

Chi-square is : 1035100.2140709655

p_value is : 0.0

degree of freedom is :1312

There is some correlation between the two variables at 0.05 significant level
```



DATA CLEANING

DATA CLEANING

CHECKING MISSING VALUES, FIND OUTLIERS, TRANSFORM VARIABLES, CREATE VARIABLES, AND DUPLICATE DATA

HANDLING Duplicate Data

```
In [21]: data=data.drop_duplicates()
```

HANDLING Missing Value

```
In [24]: #calculatin no. of missing values for each column and it's percentage
def percentage_of_miss():
    data1=data[data.columns[data.isnull().sum()>=1]] # I did slicing by condition( I get s subset of dataframe that contains co
    total_miss = data1.isnull().sum().sort_values(ascending=False)
    percent_miss = ((data1.isnull().sum()/data1.isnull().count())*100).sort_values(ascending=False) #df1.isnull().sum() returns
    missing_data = pd.concat([total_miss, percent_miss], axis=1, keys=['Number of Missing', 'Percentage'])
    return(missing_data)
```

```
In [25]: percentage_of_miss()
```

```
Out[25]:
```

	Number of Missing	Percentage
NaN	230084	100.000000
Auto-Dispatch To	187675	81.568036
Automation Name (Bot)	109317	47.511778
Automation Suite Name	109317	47.511778
Automation Type	109317	47.511778
Automation Status	109317	47.511778
Workflow State	107672	46.796822
Recovery Prime	57021	24.782688
Root Cause	13001	5.650545
Duration of Incident	9970	4.333200
Team Name	7982	3.469168

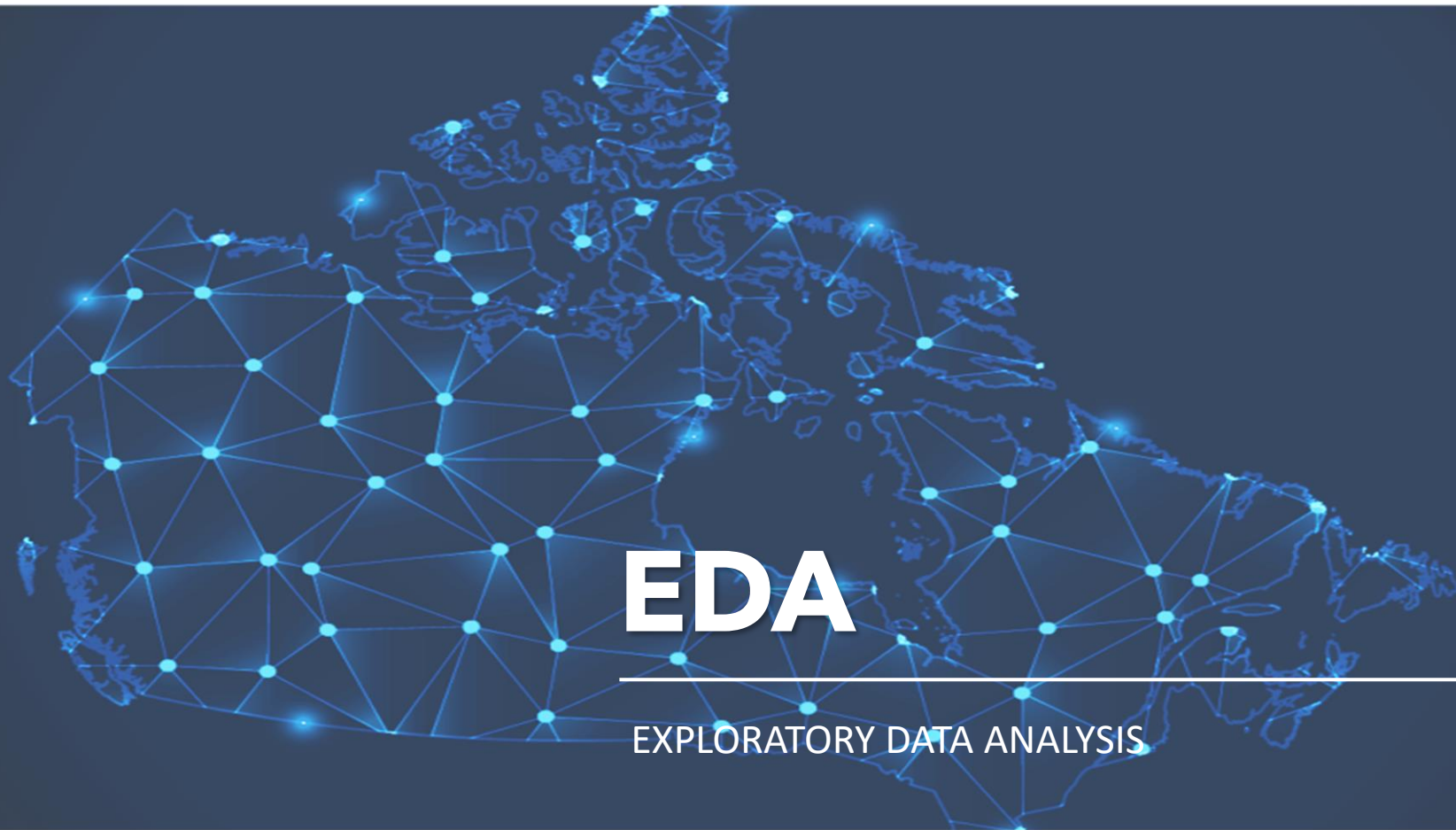
4.2.3. Drop columns that have 80% or more missing values

```
In [26]: df = data.dropna(axis=1, thresh=46016)
```

```
In [28]: # REVIEWING MISSING VALUES IN CORE DATASET - IMT
df.apply(lambda X:sum(X.isnull()))
```

```
Out[28]:
```

Imt Id	0
Create Date	0
Rule Name	0
Root Cause	0
Resolution	0
Recovery Prime	0
Workflow State	0
Incident Start	0
Incident End	0
Duration of Incident	0
Target Time To Restore	0
Event Source	0
Responsibility	0
Responsibility Department	0
State	0
Status	0
Submitter	0
Submitter Department	0
Automation Name (Bot)	0
Automation Suite Name	0
Team Name	0
Automation Type	0
Automation Status	0
Automation Framework	0
dtype: int64	



REVIEW THE DATA TYPES

DESCRIPTION

For each Tickets provided relevant information are the following:

#	Column	Non-Null Count	Dtype
0	nan	0 non-null	float64
1	Imt Id	229985 non-null	object
2	Create Date	229985 non-null	object
3	Rule Name	229985 non-null	object
4	Root Cause	216988 non-null	object
5	Resolution	226507 non-null	object
6	Recovery Prime	172968 non-null	object
7	Workflow State	122313 non-null	object
8	Incident Start	229983 non-null	object
9	Incident End	226562 non-null	object
10	Duration of Incident	220020 non-null	object
11	Target Time To Restore	228294 non-null	object
12	Event Source	229985 non-null	object
13	Responsibility	229801 non-null	object
14	Responsibility Department	229985 non-null	object
15	State	229985 non-null	object
16	Status	229985 non-null	object
17	Submitter	229985 non-null	object
18	Submitter Department	229985 non-null	object

dtypes: float64(1), object(18)
memory usage: 33.3+ MB

CONVERT TO YEAR AND MONTH

#	Column	Non-Null Count	Dtype
0	Rule Name	6762 non-null	object
1	Automation Name (Bot)	654 non-null	object
2	Automation Suite Name	654 non-null	object
3	Team Name	6762 non-null	object
4	Automation Type	654 non-null	object
5	Automation Status	654 non-null	object
6	Automation Framework	6762 non-null	object
7	Auto-Dispatch To	936 non-null	object

dtypes: object(8)
memory usage: 422.8+ KB

EXPLORATORY DATA ANALYSIS

UNDERSTANDING FEATURES

COUNTS VALUE OF ROOT CAUSE

```
In [33]: df["RootCause"].value_counts()
```

```
Out[33]: Cause Identified      35563
Software Failure      27570
Automation            27545
No Fault Found        26625
Change Management Activity  23655
Hardware Failure      19510
Cause Not Identified   17130
Unknown               13001
Commercial Power Failure 10311
Facilities - Environment  6750
Fiber                 3893
Provisioning           3569
Mother Nature         3545
Third Party           2270
Broadcast Source       1579
Customer Equipment Fault 1504
Security Issue         1088
Opened In Error        1035
Preventive Maintenance Error  993
Fiber Cut              776
OSS                    630
Coax Cable             581
Application Fault      234
Provisioning System Error 184
Switch Maintenance     164
Low RF                 115
Employee Error         80
Damage Network         80
Vandalism              60
Hosted Services        19
Client Software        11
User Access            10
Coverage               4
Name: RootCause, dtype: int64
```

PERCENTAGE OF VALUE ROOT CAUSE

```
In [34]: (df["RootCause"].value_counts(normalize=True))*100
```

```
Out[34]: Cause Identified      15.456529
Software Failure      11.982580
Automation            11.971715
No Fault Found        11.571861
Change Management Activity  10.281028
Hardware Failure       8.479512
Cause Not Identified   7.445107
Unknown               5.650545
Commercial Power Failure 4.481407
Facilities - Environment 2.933711
Fiber                 1.691991
Provisioning           1.551173
Mother Nature         1.540742
Third Party           0.986596
Broadcast Source       0.686271
Customer Equipment Fault 0.653674
Security Issue         0.472871
Opened In Error        0.449836
Preventive Maintenance Error 0.431582
Fiber Cut              0.337268
OSS                    0.273813
Coax Cable             0.252516
Application Fault      0.101702
Provisioning System Error 0.079971
Switch Maintenance     0.071278
Low RF                 0.049982
Employee Error         0.034770
Damage Network         0.034770
Vandalism              0.026077
Hosted Services        0.008258
Client Software        0.004781
User Access            0.004346
Coverage               0.001738
Name: RootCause, dtype: float64
```

CORRELATION ANALYSIS

CORRELATION BETWEEN VARIABLES

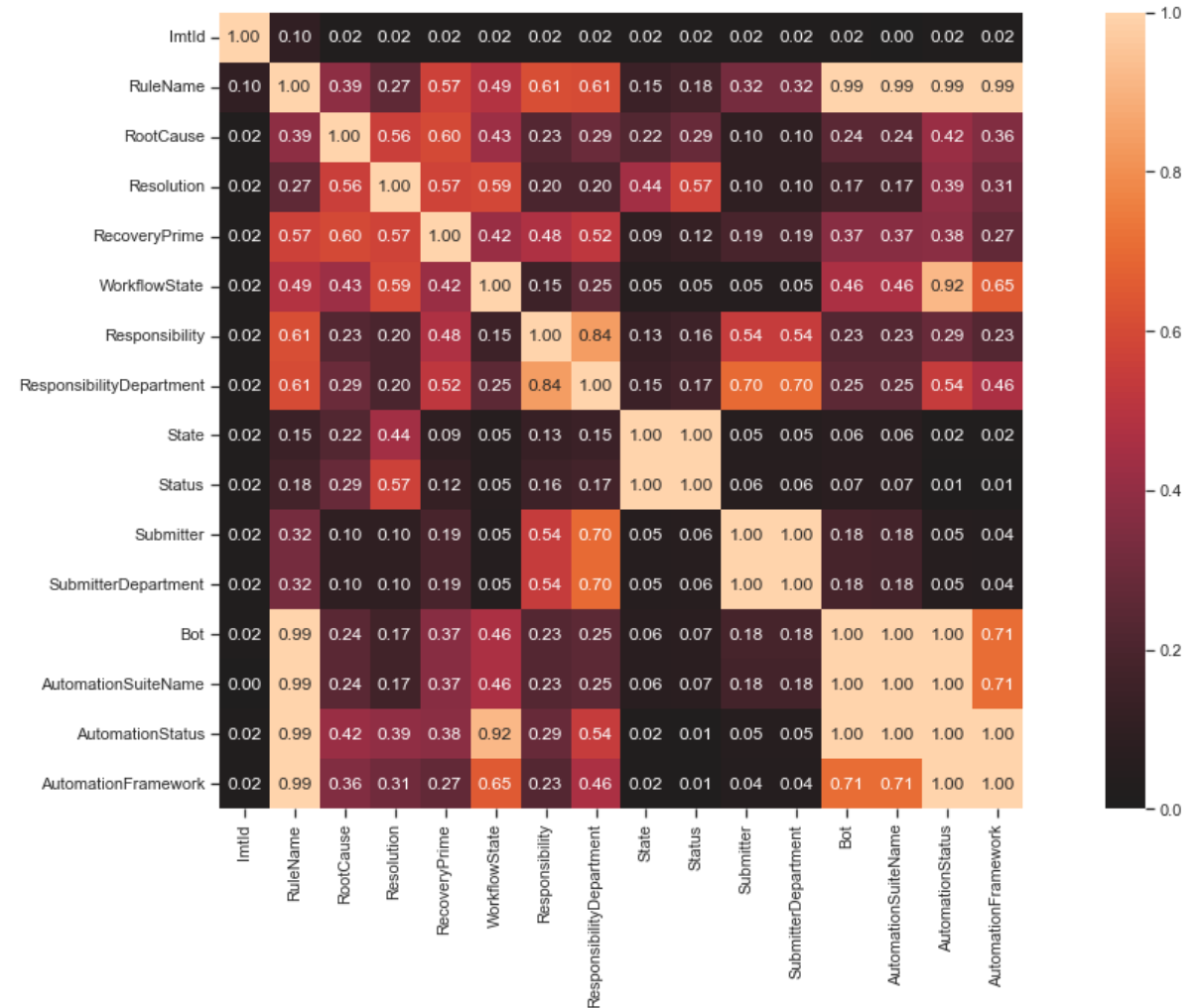
BIVARIATE ANALYSIS: **CATEGORICAL VS. CATEGORICAL**

FOR SUMMARIZATION: CONTINGENCY TABLE (TWO-WAY TABLE)

FOR VISUALIZATION :STACKED BAR CHART, GROUPED BAR CHART,...

FOR TEST OF INDEPENDENCE: CHI-SQUARE TEST

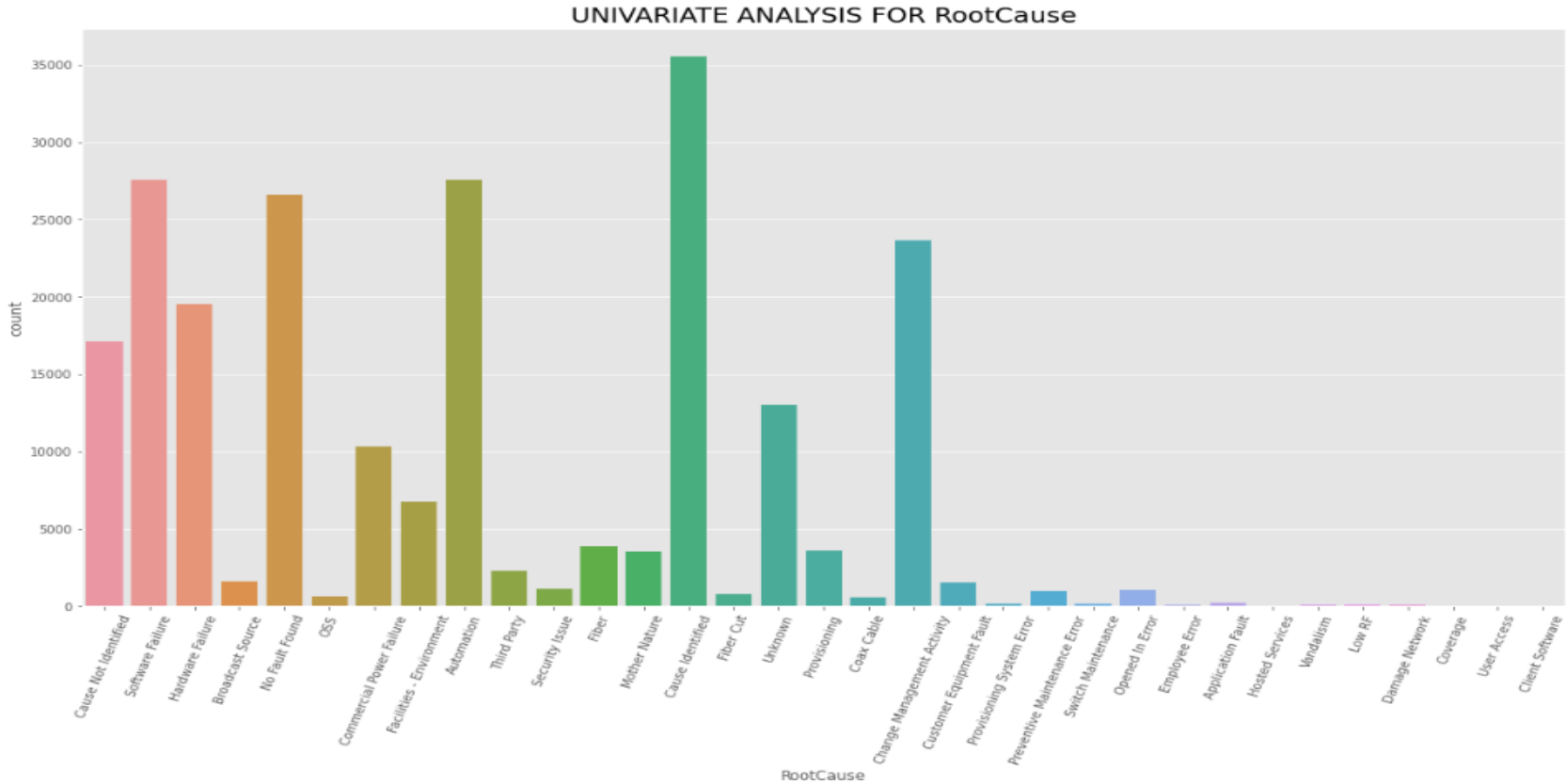
❖ **THERE IS POSITIVE CORRELATION BETWEEN ROOT CAUSE AND RECOVERY PRIME**



DATA VISUALIZATION

UNIVARIATE ANALYSIS

Visualization: BAR CHART



DATA VISUALIZATION

BIVARIATE ANALYSIS

contingency table (two-way table)

```
In [39]: print('no. of Automation')
df[df['RootCause']=='Automation'][['Bot', 'RuleName']].groupby(['Bot']).agg('count').sort_values('RuleName',
                                                    ascending=False).head(10).style.background_gradient(cmap='Wistia')
```

no. of Automation

Out[39]:

RuleName	
Bot	
DIAGNOSTIC-CASA-UPS-SWITCH-OVER	8042
DIAGNOSTIC_TWAMP	2352
DIAGNOSTIC-ARBOR-ALARM	1074
DIAGNOSTIC-IPRAN-LINK-DOWN-MAJOR-ALARMS	877
RESOLUTION-ERICSSON-LTE-SERVICE-DEGRADED	878
DIAGNOSTIC-TLAN-RECTIFIER	833
DIAGNOSTIC-IPRAN-BGP-Peer-Connection-Idle	525
DIAGNOSTIC-IPRAN-SYNTH-LINK-DOWN-CORRELATED-ALARMS	524
DIAGNOSTIC-TLAN-TEMPERATURE	410
DIAGNOSTIC-TLAN-SITE-COMMERCIAL-POWER-FAILURE	402

```
In [40]: print('no. of Automation')
df[df['RootCause']=='Automation'][['WorkflowState', 'AutomationStatus']].groupby(['WorkflowState']).agg('count').sort_values(
                                                    ascending=False).head(10).style.background_gradient(cmap='Wistia')
```

no. of Automation

Out[40]:

Automation Status	
Workflow State	
Closed by Auto Diag	15063
Closed by Auto Res/Val	11



FEATURE ENGINEERING

PREDICTING INDEPENDENT VARIABLE

FEATURE ENGINEERING

MEMORY REDUCTION

FEATURE ENGINEERING: MEMORY REDUCTION

```
In [132]: def reduce_mem_usage(df):
          """ iterate through all the columns of a dataframe and modify the data type
              to reduce memory usage.
          """
          start_mem = df.memory_usage().sum() / 1024 ** 2
          print('Memory usage of dataframe is {:.2f} MB'.format(start_mem))

          for col in df.columns:
              col_type = df[col].dtype

              if col_type != object:
                  c_min = df[col].min()
                  c_max = df[col].max()
                  if str(col_type)[:3] == 'int':
                      if c_min > np.iinfo(np.int8).min and c_max < np.iinfo(np.int8).max:
                          df[col] = df[col].astype(np.int8)
                      elif c_min > np.iinfo(np.int16).min and c_max < np.iinfo(np.int16).max:
                          df[col] = df[col].astype(np.int16)
                      elif c_min > np.iinfo(np.int32).min and c_max < np.iinfo(np.int32).max:
                          df[col] = df[col].astype(np.int32)
                      elif c_min > np.iinfo(np.int64).min and c_max < np.iinfo(np.int64).max:
                          df[col] = df[col].astype(np.int64)
                  else:
                      if c_min > np.finfo(np.float16).min and c_max < np.finfo(np.float16).max:
                          df[col] = df[col].astype(np.float16)
                      elif c_min > np.finfo(np.float32).min and c_max < np.finfo(np.float32).max:
                          df[col] = df[col].astype(np.float32)
                      else:
                          df[col] = df[col].astype(np.float64)
              else:
                  df[col] = df[col].astype('category')

          end_mem = df.memory_usage().sum() / 1024 ** 2
          print('Memory usage after optimization is: {:.2f} MB'.format(end_mem))
          print('Decreased by {:.1f}%'.format(100 * (start_mem - end_mem) / start_mem))

          return df
```



MODEL GENERATION

SUPERVISED MACHINE LEARNING CLASSIFIER

SUPERVISED MACHINE LERNING CLASSIFIER

VARIABLE SELECTION

Defining 2 dataset :
Dependent variable (y)
Independent variables (X)

```
In [64]: cols = ['RootCause', 'ImtId', 'CreateDate', 'RuleName', 'Resolution', 'RecoveryPrime', 'WorkflowState',  
                'Responsibility', 'ResponsibilityDepartment', 'State', 'Status', 'Submitter', 'SubmitterDepartment',  
                'Bot', 'AutomationSuiteName', 'AutomationStatus', 'AutomationFramework']
```

```
#Splitting dataset into Labels and features  
X = df.drop(columns = cols)  
y = df.RootCause_NUM
```

```
In [65]: #Splitting dataset into Labels and features  
X = df.drop(columns = ['RootCause_NUM', 'RootCause', 'ImtId', 'CreateDate', 'RuleName', 'Resolution', 'RecoveryPrime', 'WorkflowSta',  
                      'Responsibility', 'ResponsibilityDepartment', 'State', 'Status', 'Submitter', 'SubmitterDepartment',  
                      'Bot', 'AutomationSuiteName', 'AutomationStatus', 'AutomationFramework'])  
y = df.RootCause_NUM
```

```
In [66]: X.head()
```

Out[66]:

	ImtId_NUM	CreateDate_NUM	RuleName_NUM	Resolution_NUM	RecoveryPrime_NUM	WorkflowState_NUM	Responsibility_NUM	ResponsibilityDepartment
0	0	0	1	1	2	6	22	
1	1	1	1	1	3	6	15	
2	2	2	1	1	2	6	21	
3	3	3	1	1	3	6	15	
4	4	4	1	1	3	5	15	

```
In [67]: y.head()
```

Out[67]:

```
0    3  
1    5  
2    5  
3    1  
4    5  
Name: RootCause_NUM, dtype: int64
```

```
In [6]: df.rename(columns=df.iloc[0], inplace = True)  
df.drop([0], inplace = True)  
df
```

Out[6]:

	NaN	Imt Id	Create Date	Rule Name	Root Cause	Resolution	Recovery Prime	Workflow State	Incident Start	Incident End	Duration of Incident	Target Time To Restore
1	NaN	IMT.2112.043195	2022-01-01 00:02:31	UMTS UL RSSI -85dBm	Cause Not Identified	Network / Service Validated	Field Resolved	NaN	2022-01-01 00:00:03	2022-01-11 16:24:39	10 days, 16 Hrs, 24 Minutes	2022-01-08 00:00:03
2	NaN	IMT.2112.043196	2022-01-01 00:03:41	BMC - IUM Evolution - Parameter Count changed ...	Software Failure	Network / Service Validated	NOC Resolved	NaN	2021-12-31 23:50:35	2022-01-01 00:37:21	0 days, 0 Hrs, 46 Minutes	2022-01-03 23:50:35

SUPERVISED MACHINE LERNING CLASSIFIER

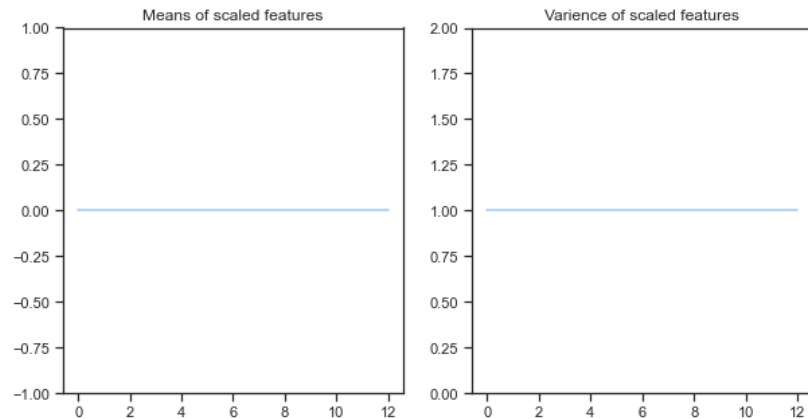
VARIABLE STANDARDIZED

GETTING VARIANCE AND STANDARDIZED

```
In [91]: #check weathear data is standardized or not
plt.subplot(121)
plt.ylim(-1,1)

means=[]
for i in range(X_scaled.shape[1]):
    means.append(np.mean(X_scaled.iloc[:,i]))
plt.plot(means, scaley=False)
plt.title('Means of scaled features')

plt.subplot(122)
plt.ylim(0,2)
vars=[]
for i in range(X_scaled.shape[1]):
    vars.append(np.var(X_scaled.iloc[:,i]))
plt.plot(vars, scaley=False)
plt.title('Variance of scaled features')
plt.show()
```



$$\sigma^2 = \frac{\sum (x - \mu)^2}{N} \quad \text{Population Variance}$$

$$s^2 = \frac{\sum (x - \bar{x})^2}{n - 1} \quad \text{Sample Variance}$$

$$s = \sqrt{\frac{\sum (x - \bar{x})^2}{n - 1}} \quad \text{Sample Standard Deviation}$$

SUPERVISED MACHINE LEARNING CLASSIFIER

KNN

```
*****.KNN Model*****
```

```
In [96]: from sklearn.neighbors import KNeighborsClassifier
```

```
In [97]: knn = KNeighborsClassifier()
knn.fit(X_train,y_train)
pred_knn = knn.predict(X_test)
score_knn = accuracy_score(y_test,pred_knn)
```

```
In [98]: print('confusion_matrix KNN :\\n', metrics.confusion_matrix(y_test, pred_knn))
print('-----')
print('classification_report KNN :\\n',metrics.classification_report(y_test, pred_knn))
```

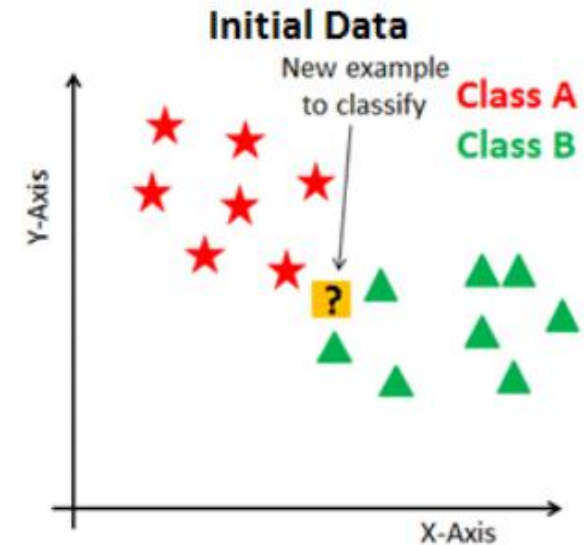
```
confusion_matrix KNN :
[[ 5711   1  121   74   67 1049]
 [    0  122  107    4    4  151]
 [  278   62 4726  305  651 2796]
 [  239   10  661  967  480 1947]
 [  157    3  908  214 2432 2168]
 [ 1550   90 3091  978 1968 23429]]

-----

classification_report KNN :
              precision    recall  f1-score   support

    0             0.72       0.81       0.76       7023
    1             0.42       0.31       0.36        388
    2             0.49       0.54       0.51       8818
    3             0.38       0.22       0.28       4304
    4             0.43       0.41       0.42       5882
    5             0.74       0.75       0.75      31106

 accuracy              0.65       57521
 macro avg             0.53       0.51       0.52       57521
 weighted avg          0.64       0.65       0.64       57521
```



SUPERVISED MACHINE LERNING CLASSIFIER

KNN-OPTIMIZED

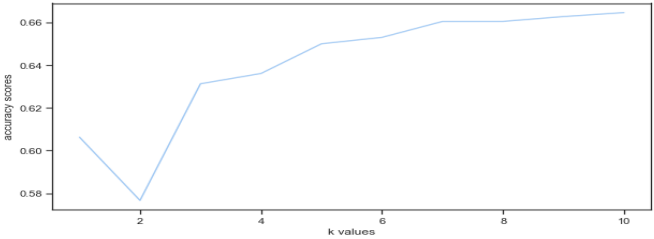
2.3.1. KNN Opt

```
*****.....KNN Opt.....*****

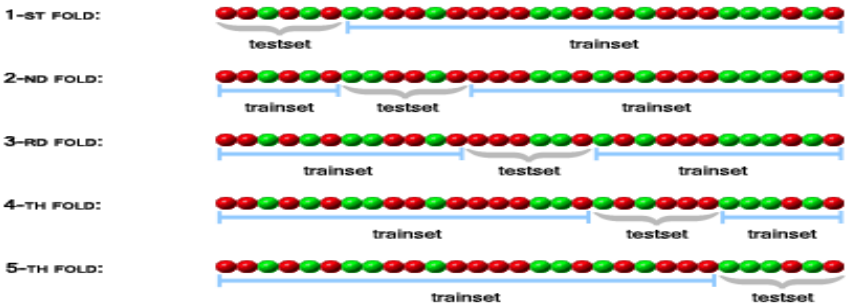
In [117]: from sklearn.neighbors import KNeighborsClassifier
          from sklearn.metrics import accuracy_score
          list_knn=[]
          for i in range(1,11):
              knn = KNeighborsClassifier(n_neighbors = i)
              knn.fit(X_train,y_train)
              pred_s = knn.predict(X_test)
              scores = accuracy_score(y_test,pred_s)
              list_knn.append(scores)

In [118]: print(max(list_knn))
          0.6645572921193998

In [119]: import matplotlib.pyplot as plt
          plt.plot(range(1,11),list_knn)
          plt.xlabel('k values')
          plt.ylabel('accuracy scores')
          plt.show()
```



ONE ITERATION OF A 5-FOLD CROSS-VALIDATION:



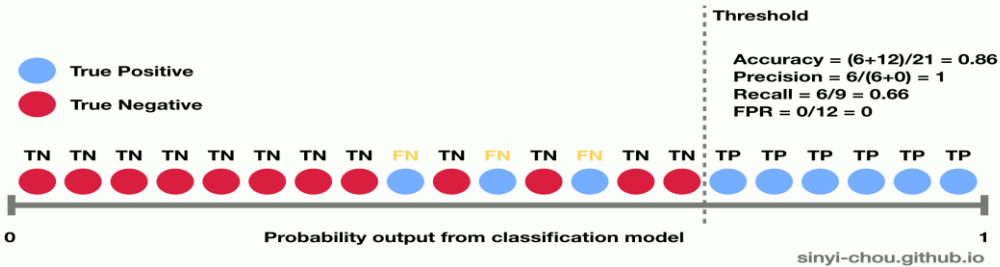
```
In [122]: print('confusion_matrix KNN_opt :\\n', metrics.confusion_matrix(y_test, pred_knn_g))
          print('-----\\n')
          print('classification_report KNN_opt :\\n',metrics.classification_report(y_test, pred_knn_g))
```

```
confusion_matrix KNN_opt :
```

[[5755	1	90	62	48	1067]	
[0	109	108	1	1	169]
[299	63	4586	168	534	3168]
[243	13	609	847	391	2201]
[163	5	813	152	2239	2510]
[1592	85	2650	572	1517	24690]]

classification_report KNN_opt :

	precision	recall	f1-score	support
0	0.71	0.82	0.76	7023
1	0.39	0.28	0.33	388
2	0.52	0.52	0.52	8818
3	0.47	0.20	0.28	4304
4	0.47	0.38	0.42	5882
5	0.73	0.79	0.76	31106
accuracy			0.66	57521
macro avg	0.55	0.50	0.51	57521
weighted avg	0.65	0.66	0.65	57521





CONCLUSION

DECODING INFERENCE

CONCLUSION

BEST MODEL

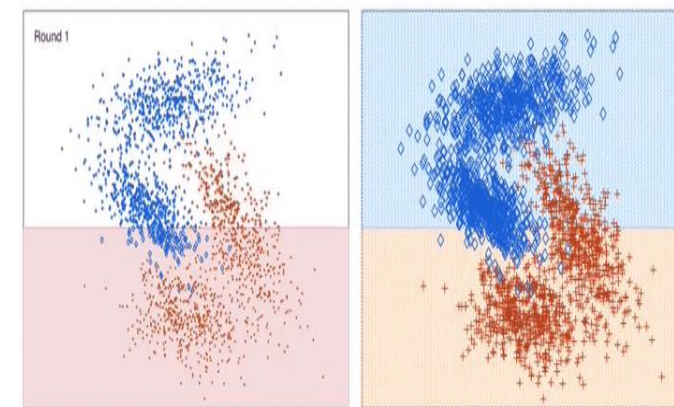
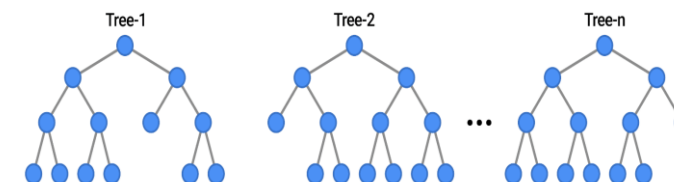
B. COMPARE ALL MODELS

```
In [139]: print('Display Accuracy, Recall and Precision For Classification Models')
print('-----\n')
print('-----')
print('*****INFERENCE AND CONCLUSION*****')
print('-----')
results_df = append_results("KNN", knn, results_df, y_test, pred_knn)
results_df = append_results("Logestic Regression", log_reg, results_df, y_test, pred_log)
#results_df = append_results("SVC", svc, results_df, y_test, pred_svc)
results_df = append_results("Random Forest", rfc, results_df, y_test, pred_rfc)
results_df = append_results("Ada Boost", adab, results_df, y_test, pred_adab)
results_df = append_results("KNN_opt", grid_model, results_df, y_test, pred_knn_g)
results_df = append_results("LogesticRegression_opt", grid_model2, results_df, y_test, pred_log_g)
#results_df = append_results("SVC_opt", grid_model3, results_df, y_test, pred_svc_g)
results_df = append_results("Random Forest_opt", grid_model4, results_df, y_test, pred_rfo)
results_df = append_results("Ada Boost_opt", grid_model5, results_df, y_test, pred_adab_o)
results_df
```

Out[139]:

	Model	Cross Val Score	pre_macro	recall_macro	f1_macro	f1_macro_manual
0	KNN	0.742112	0.707708	0.722747	0.722747	0.939
1	Logestic Regression	0.545525	0.515497	0.484102	0.484102	0.942
2	Random Forest	0.784748	0.772128	0.777991	0.777991	0.946
3	Ada Boost	0.411625	0.499071	0.412114	0.412114	0.938
4	KNN_opt	0.747072	0.738130	0.742032	0.742032	0.940
5	LogesticRegression_opt	0.349454	0.332449	0.294926	0.294926	0.940
6	Random Forest_opt	0.886772	0.564079	0.578637	0.578637	0.948
7	Ada Boost_opt	0.905025	0.894448	0.899432	0.899432	0.933

EXAMPLES



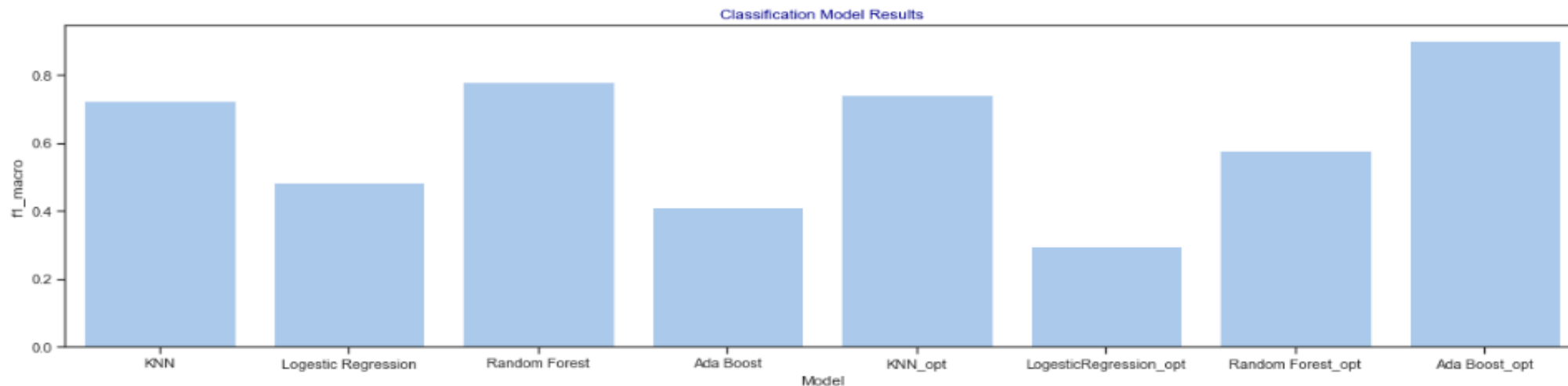
CONCLUSION

BEST MODEL

The model is acceptable because with an accuracy of over (60%)

A different model (non-linear) will be better to predict the output variable as the correlation study suggests.

```
In [140]: ▶ plt.rcParams['figure.figsize'] = 20,5  
g = sns.barplot("Model", "f1_macro", data = results_df, color = 'b')  
g.set_title("Classification Model Results", color = "darkblue")  
plt.show()
```



INFERENCE

INTERPRETING THE RESULTS

Data Prediction based on data history

After selecting the model, data analysis is performed based on the predicted data.

```
In [54]: # Defining all the conditions inside a function
def condition(x):
    if x == 'Automation':
        return 0
    elif x == 'Broadcast Source':
        return 1
    elif x == 'Cause Identified':
        return 2
    elif x == 'Cause Not Identified':
        return 3
    elif x == 'Change Management Activity':
        return 4
    else:
        return 5

# Applying the conditions
df['RootCause_NUM'] = df['RootCause'].apply(condition)

print(df['RootCause_NUM'])
```

```
0      3
1      5
2      5
3      1
4      5
..
96996  5
96997  5
96998  5
96999  5
97000  5
Name: RootCause_NUM, Length: 97100, dtype: int64
```

```
In [148]: predDf = pd.DataFrame({'RuleName_NUM':df_RuleName_M, 'RootCause_NUM':pred_adab_o})
predDf
```

Out[148]:

	RuleName_NUM	RootCause_NUM
0	UMTS UL RSSI -85dBm	5.0
1	BMC - IUM Evolution - Parameter Count changed ...	5.0
2	TLAN_HVAC_Failure	2.0
3	SB_SYNTH_IPTV - Medius - Critical	0.0
4	CCAP - CASA System Monitoring UPS Switch over	4.0
...

```
In [150]: predDf = pd.DataFrame({'RecoveryPrime_NUM':df_RecoveryPrime_M, 'RootCause_NUM':pred_adab_o})
predDf
```

Out[150]:

	RecoveryPrime_NUM	RootCause_NUM
0	Field Resolved	5.0
1	NOC Resolved	5.0
2	Field Resolved	2.0
3	NOC Resolved	0.0
4	NOC Resolved	4.0
...



RECOMMENDATION

RECOMMENDATION

PREDICTIVE MAINTENANCE AND IMPROVE NETWORK OPTIMIZATION

Total IMT in the ESAP for **PY** : 283033 , **Average in a Month: 23586**
Total IMT for PY: 424565

Total IMT in the ESAP for **YTD** : 245743, **Average in a Month: 31709**
Total IMT for YTD: 355573

1

Problem Statement

- Detect Network Congestion (Network Monitoring & Network Device)
- Prevent power outages, by using AI/ML Predictive Maintenance, and make customer satisfaction.
- Identify Network Problems and issues fast by using ML



2

Business Value

- According to the forecast, we can reduce the time, which leads to saving money and improve business processes, and make customer satisfy.
- By prediction, it is possible to determine the workload of network teams, which leads to handling or the new definition of projects in the system,



3

Data

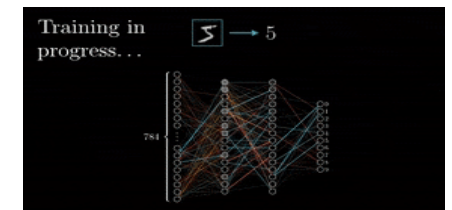
- Enable cross team collaboration
- Identified Gaps and Limitations such as data and cloud environment



4

Model Generation

- Using High level Model
- Due to many fluctuated in top Flow and Rule Name, we need to use algorithms that are suitable for getting the bias.



Thank You !

SARA!



JUST FOR YOUR INFORMATION

NETWORK

Current mode: **Modify**

Save

New search

My Reports

Advanced search

Clear

Logout

Help

Home

1 of 1 results

Preferences

Refresh

IMT ID	Workflow ...	Incident Start	Network Impact	Service Impact	Customer Impact	Severity	State	Platform	Headline	Assigned To	Target Time
IMT.2208.019730	Closed by Aut	8/16/2022 12:03:42 A	Threatened	None	Medium	P4	Closed	Core Data	--- () - dp01.ym - dp01.ym	ESAP Automation	8/23/20

Report

Select All

DeSelect All

Number 1 of 1 | 1 row selected

IMT ID

IMT.2208.019730

Create Date

8/16/2022 12:06:12 AM

Core Data

Core Security - peakflowSPTTraps CRITICAL - Core - Security - dp01.ym - Signature exceeding threshold: Total Traffic - MQ: "Auto Mitigation of Rogers HSI Blocks MQ"

Network Layer

Core

Severity

P4

None

Brand Impact

No

Customer Impact

Medium

Submitter: ESAP

Escalations

Details / Print

Progress

Tasks

IPTV

Notifications

Elements

Services

Logs

Events

Queue

Queue Type

Ticket Type

Associated Ticket

CTCount

RMA

Caused by a NCT

Open

1 entries returned - 1 entries matched

Type	Associated Ticket
ESAP	AUTO.10348982

Incident Start

8/16/2022 12:03:42 AM

Incident End

8/16/2022 12:12:20 AM

Duration of Incident

0 days, 0 hours and 9 minutes

Target Time To Restore

8/23/2022 12:03:42 AM

Workflow State

Closed by Auto Diag

Main Element

Progress Information

Primary Element Type

Restorative Technician

Root Cause

Automation

Resolution

Network/Service Validation

Recovery Prime

Automation

TBS Update

Yes

Unplanned due to Change

Cust Ticket?

False

Event Source

ESAP-AUTO

Site Console

Map

HFC

0 entries returned - 0 entries matched

Reference

JUST FOR YOUR **INFORMATION**

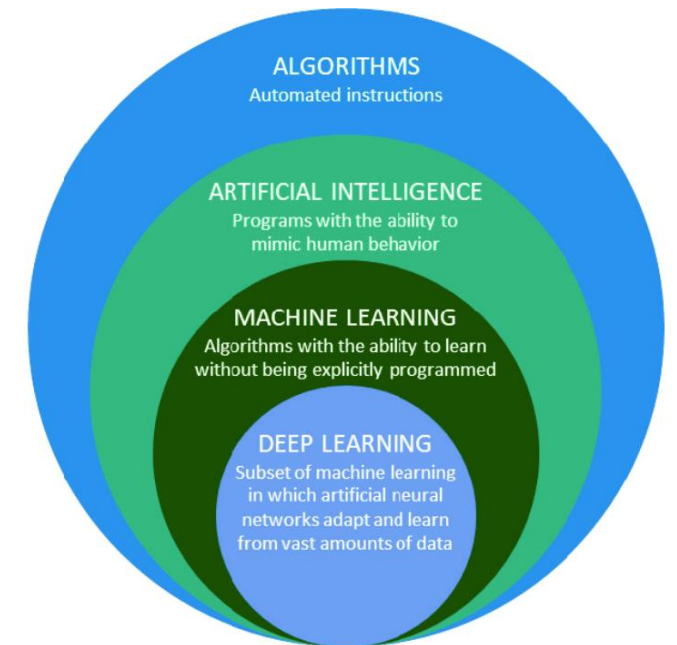
NETWORK

WHEN DON'T WE NEED AI AND ML?

- WHEN YOU ARE LOOKING FOR A SOLUTION THAT CAN BE MEMORIZED.
- WHEN THERE IS NO PATTERN IN YOUR DATA.
- WHEN WE DON'T HAVE ENOUGH DATA OR ACCURATE DATA.
- WHEN IT IS NOT FEASIBLE TO EXECUTE OR UPDATE THE SOLUTION.
- WHEN ML AND AI IS NOT EXPLAINABLE.
- WHEN ML AND AI IS BIASED.

WHEN DO WE NEED AI AND ML?

- WHEN THERE IS AN EXISTING BENCHMARK TO VALIDATE PERFORMANCE.
- WHEN THERE IS RELIABLE DATA TO TRAIN MODEL.
- WHEN THERE IS A KNOWN METRIC TO MEASURE PERFORMANCE.



JUST FOR YOUR **INFORMATION**

NETWORK

**Before any ML project, we need you to...
Before any ML project we need you to...**

Problem Statement

What are the challenges/problems you are currently facing?
If there was no limitation, how does the ideal solution look like for you?
How are you currently trying to address the problem?
Does a rule-based model already exist? Please share the rules?
What is an appropriate ML/AI solution for this challenge?
Do you have the data/stats that supports your theory? Where can it be found?
What is the motivation of gathering this data (who created, purpose, who funded, any comment?)
Validating the solution - if we could solve the problem with ML and AI, what Rogers can do with this information?

Business Value

Quantify the business problem: how would this solution save money or improve business processes?
What is the direct and tangible outcomes as the result of using this solution?
What do we define as success in terms of metrics after implementing this solution? What KPIs we are aiming?
What are the trade-offs? Does the effort of building and maintaining the AI/ML solution worth the costs we are aiming to solve?
What would be the dollar value of this solution to the business after implementation

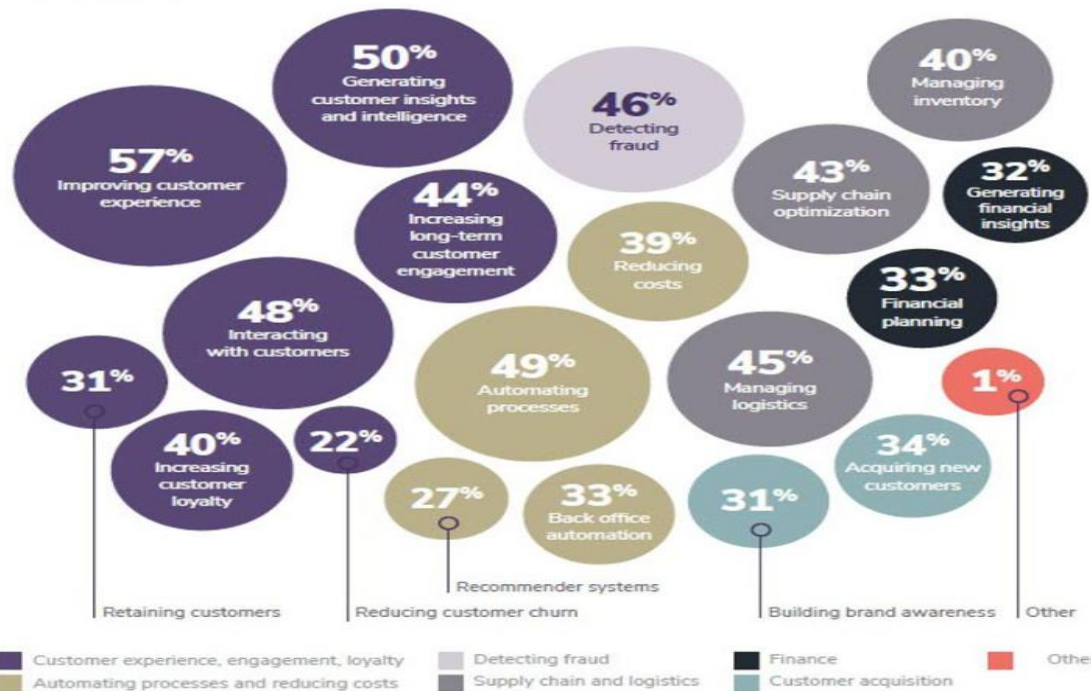
Data Understanding

Do you have data accessible?
What is the composition of data? (data type, data sample, instances, columns names, dictionary, missing information, any relationship between columns, interdependencies to other data, data behavior)
What is the collection process? (How, who? Mechanism -manual or automatic, review process, cleaning process?)
Use cases? Has anyone using the data?
Preprocessing and cleaning steps? Software's to process data?
How is this distributed? Copyright? privacy concern? Regulation of access? Third part involved?
Data maintenance? Who hosts data? Limitation?

JUST FOR YOUR **INFORMATION**

NETWORK

Customer experience and process automation represent the top AI/ML use cases



ALGORITHMIA'S THIRD ANNUAL SURVEY, 2021 ENTERPRISE TRENDS IN MACHINE LEARNING

□ Using PyTorch, TensorFlow, Keras Spark, and NLTK in the next step

The Formula for Chi Square Is

$$\chi^2_c = \sum \frac{(O_i - E_i)^2}{E_i}$$

where:

c = degrees of freedom

O = observed value(s)

E = expected value(s)