

Data Science PROJECT

Client: ABC Tech | Category: ITSM - ML

Project Ref: PM-PR-0012

Business Case:

ABC Tech is a mid-size organisation operating in the IT-enabled business segment over a decade. On an average ABC Tech receives 22-25k IT incidents/tickets, which were handled to best practice ITIL framework with incident management, problem management, change management and configuration management processes. These ITIL practices attained a matured process level and a recent audit confirmed that further improvement initiatives may not yield a return on investment.

ABC Tech management is looking for ways to improve the incident management process as recent customer survey results show that incident management is rated as poor.

Machine Learning as a way to improve ITSM processes ABC Tech management recently attended a Machine Learning conference on ML for ITSM.

Machine learning looks prospective to improve ITSM processes through prediction and automation. They came up with 4 key areas, where ML can help ITSM process in ABC Tech.

1. Predicting High Priority Tickets: To predict priority 1 & 2 tickets, so that they can take preventive measures or fix the problem before it surfaces.
2. Forecast the incident volume in different fields, quarterly and annual. So that they can be better prepared with resources and technology planning.
3. Auto tag the tickets with right priorities and right departments so that reassigning and related delay can be reduced.
4. Predict RFC (Request for change) and possible failure / misconfiguration of ITSM assets.

Data Set Fields:

Total of about 46k records from year 2012,2013,2014

Data needs to be queried from MYSQL data base (Read Only Access)

Host: 18.136.56.185

Port: 3306

Username : dm_team

Password: dm_team123#

CI_Name	SUB000508
CI_Cat	subapplication
CI_Subcat	Web Based Application
WBS	WBS000162
Incident_ID	IM0000004
Status	Closed
Impact	4
Urgency	4
Priority	4
Category	incident
KB_number	KM0000553
Alert_Status	closed
No_of_Reassignments	26
Open_Time	05/02/2012 13:32:57
Reopen_Time	
Resolved_Time	04/11/2013 13:50:27
Close_Time	04/11/2013 13:51:17
Handle_Time_hrs	3871,691111
Closure_Code	Other
No_of_Related_Interactions	1
Related_Interaction	SD0000007
No_of_Related_Incidents	2
No_of_Related_Changes	1
Related_Change	C00000056

CI_Name : Configuration Item Number

CI_Cat : Configuration Item Category

CI_Subcat : Configuration Item sub-category

The above three fields are provided by Configuration Management

Database(CMDB) https://en.wikipedia.org/wiki/Configuration_management_database

WBS : Work breakdown structure - A work-breakdown structure in project management and systems engineering, is a deliverable-oriented breakdown of a project into smaller components. https://en.wikipedia.org/wiki/Work_breakdown_structure

Incident_ID : Incident Identification

Who detected the incident and how? How soon was the incident detected after it occurred? Could the incident have been identified earlier? Could any tools or technologies have aided in the prompt or pre-emptive detection of the incident? <https://www.manageengine.com/products/service-desk/itil-incident-management-guide.html>

Status : Status of the item whether it is open or closed.

Impact : Impact is a measure of the effect of an incident, problem, or change on business processes. Impact is often based on how service levels will be affected.

Urgency : Urgency is a measure of how long it will be until an incident, problem, or change has a significant business impact. For example, a high impact incident may have low urgency if the impact will not affect the business until the end of the financial year.

Priority : Priority is a category that identifies the relative importance of an incident, problem, or change. Priority is based on impact and urgency, and it identifies required times for actions to be taken. Impact and urgency are used to assign priority.

<https://docs.bmc.com/docs/display/public/rondsubscriber/Impact%2C+urgency%2C+and+priority+criteria#:~:text=Impact%20is%20a%20measure%20of%20has%20a%20significant%20business%20impact>.

Category : Whether the ticket is incident or request for information

KB_number : Knowledge base number

Alert_Status : Status of the ticket whether it is closed or not closed

No_of_Reassignments : Number of reassigned tickets

Open_Time : Time of opening

Reopen_Time : Time of re-opening

Resolved_Time : Time of resolving

Close_Time : Time of closing

Handle_Time_hrs : Time of handling the incidents

Closure_Code : Code of closure

No_of_Related_Interactions : Number of related interactions

Related_Interaction : Use of a separate Interaction Management process allows for a separation of Incident metrics from Requests for Information.

No_of_Related_Incidents : Number of related incidents

No_of_Related_Changes : Number of related changes

Related_Change : A change request is a document containing a call for an adjustment of a system.

The goal of this project is to improve the ITSM process.

5. Features

The feature of the datasets were provided by ***Datamites*** company.

PRIORITY Matrix

		Urgency					
		1	2	3	4	5	5 - very low
Impact	1	1	2	3	3	3	3
	2	2	2	2	3	3	4
	3	2	2	3	3	4	4
	4	3	3	3	4	4	4
	5	3	3	4	4	5	5

ITSM Description

IT service management (ITSM) refers to the entirety of activities – directed by policies, organized and structured in processes and supporting procedures – that are performed by an organization to design, plan, deliver, operate and control information technology (IT) services offered to customers. https://en.wikipedia.org/wiki/IT_service_management.



Figure 1: ITSM diagram

Assumptions:

- Used forward fill and backward fill to fill in the null values.
- For the first part drop the following fields: 'Urgency','Impact','Alert_Status','No_of_Related_Incidents','No_of_Related_Changes'.
- For the first part use priority as target variable.

- For second part, used Open_Time and Incident_ID fields for forecasting incidents using ARIMA and SARIMA
- Used Rolling forecast on ARIMA and SARIMA model to improve the predicted data.
- Used more data in train dataset than the test dataset in forecasting incidents.
- For the third part, drop the following fields: 'Urgency','Impact','Alert_Status','Open_Time','Reopen_Time','Close_Time','Resolved_Time'
- Used No_of_Reassignments as target variable for finding the tickets with right priorities and right departments.
- For the fourth part, drop the following fields: 'Urgency','Impact','Alert_Status','No_of_Related_Incidents','Status','Open_Time','Reopen_Time','Close_Time','Resolved_Time'
- Used No_of_Related_Changes as target variable for predicting RFC (Request for change) and possible failure misconfiguration of ITSM assets.

Import the dataset from the server

```
!pip install sqlalchemy
!pip install pymysql
!pip install --upgrade pip
!pip install imblearn
from sqlalchemy import create_engine
import pandas as pd
db_host= '18.136.56.185:3306'
username = 'dm_team'
user_pass= 'dm_team123#'
db_name='project_itsm'
conn=create_engine('mysql+pymysql://dm_team:dm_team123#@18.136.56.185:3306/project_itsm')
conn.table_names()
query = 'select * from dataset_list '
dataset_list = pd.read_sql(query,conn)
print(dataset_list .shape)
dataset_list
```

Download the dataset in csv format

```
dataset_list.to_csv('C:\\Users\\DELL\\Desktop\\Rubixe projects\\Mar2020\\ITSM_data.csv')
```

Approach:

1. **Predicting High Priority Tickets:** To predict priority 1 & 2 tickets, so that they can take preventive measures or fix the problem before it surfaces.

1) Import the necessary packages

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams
%matplotlib inline
from collections import Counter
import warnings
warnings.filterwarnings("ignore")
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
```

```

from sklearn.preprocessing import LabelEncoder,scale
from sklearn.metrics import
accuracy_score,precision_score,confusion_matrix,classification_report,f1_score,recall_score
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA

```

2) Load the dataset

```

data_parser=lambda c: pd.to_Dataframe(c,format='%d/%m/%Y %H:%M:%s')
data=pd.read_csv('C:\\Users\\DELL\\Desktop\\Rubixe           projects\\Mar2020\\ITSM_data.csv',
parse_dates=['Open_Time','Reopen_Time','Close_Time','Resolved_Time'])

```

3) Checking for outliers

```

sns.set_style('whitegrid')
sns.heatmap(data.isnull(),yticklabels=False,cbar=True,cmap='viridis')

```

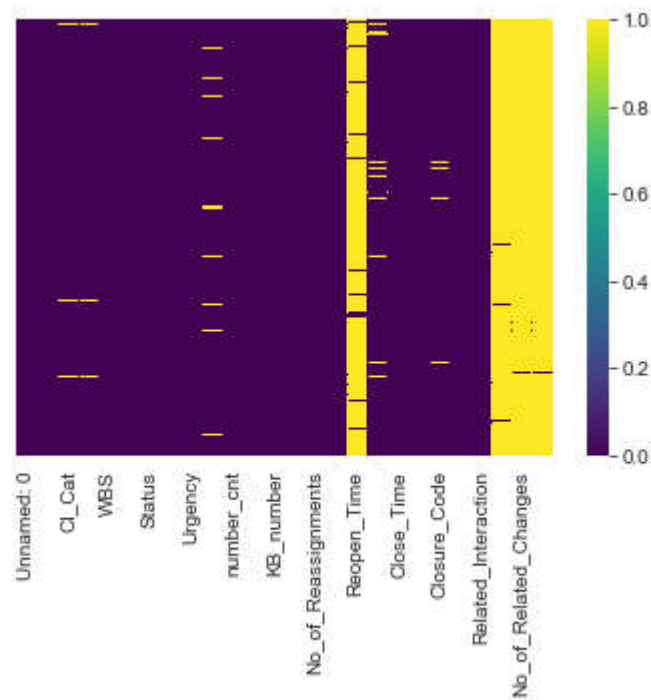


Figure 2: Heatmap to detect outliers in the data

```

data[['Priority']].boxplot();

```

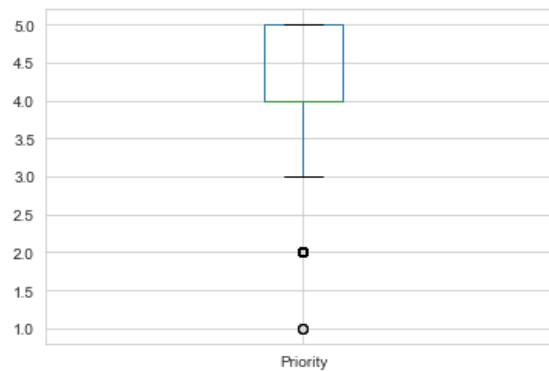


Figure 3: Outliers using boxplot

```
X=data.Priority
print(str(X[removed_outliers_Priority].size)+"/"+str(X.size)+" data points remain")
plt.boxplot(X[removed_outliers_Priority]);
plt.xlabel("Priority")
```



Figure 4: Removed outliers

```
figure,axis=plt.subplots(1,2,figsize=(16,5))
axis[0].boxplot(X);
axis[1].boxplot(X[removed_outliers_Priority]);
axis[0].set_title("With outliers")
axis[0].set_xlabel("Priority")
axis[1].set_title("Removed outliers")
axis[1].set_xlabel("Priority")
```


Out[284]: Text(0.5, 0, 'Priority')



Figure 5: Boxplots with outliers and after removing outliers

```
data['clean_Priority']=X[removed_outliers_Priority]
```

4) Check for EDA

```
data.info()
data.describe().T
data.shape
data.head()
data.isnull().sum().to_frame().T
data.fillna(method='ffill',inplace=True)
data.isnull().sum().to_frame().T
data.fillna(method='bfill',inplace=True)
data.isnull().sum().to_frame()
Counter(data.clean_Priority)
Drop the fields:
data_new=data.drop(['Urgency','Impact','Alert_Status','No_of_Related_Incidents','No_of_Related_Changes','Related_Change','Reopen_Time','Priority'],axis=1)
data_new.head()
```

5) Define X and y

```
X=data_new.iloc[:,data_new.columns!='clean_Priority']
y=data_new.clean_Priority
X.info()
y.describe().T
```

6) Using Label Encoder

```
enc=LabelEncoder()
X.CI_Name=enc.fit_transform(X.CI_Name)
X.CI_Cat=enc.fit_transform(X.CI_Cat)
X.CI_Subcat=enc.fit_transform(X.CI_Subcat)
X.Status=enc.fit_transform(X.Status)
X.Closure_Code=enc.fit_transform(X.Closure_Code)
X.Category=enc.fit_transform(X.Category)
X.WBS=enc.fit_transform(X.WBS)
X.Incident_ID=enc.fit_transform(X.Incident_ID)
```

```

X.Related_Interaction=enc.fit_transform(X.Related_Interaction)
X.KB_number=enc.fit_transform(X.KB_number)
X.Open_Time=enc.fit_transform(X.Open_Time)
X.Resolved_Time=enc.fit_transform(X.Resolved_Time)
X.Close_Time=enc.fit_transform(X.Close_Time)
X.Handle_Time_hrs=enc.fit_transform(X.Handle_Time_hrs)
X.number_cnt=enc.fit_transform(X.number_cnt)
X.No_of_Related_Interactions=enc.fit_transform(X.No_of_Related_Interactions)
X.No_of_Reassignments=enc.fit_transform(X.No_of_Reassignments)
X.head()
X.info()

```

6) Train-test split

```

X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=20,test_size=0.3) print("X_train shape
=",X_train.shape)
print("X_test shape = ",X_test.shape)
print("y_train shape = ",y_train.shape)
print("y_test shape = ",y_test.shape)

```

7) Random-Forest Classifier

```

Define the model
model=RandomForestClassifier(n_estimators=250,random_state=10,criterion='gini')
model.fit(X_train,y_train)
print(model.feature_importances_)
pd.DataFrame(model.feature_importances_,index=X.columns).sort_values(0,ascending=False)
y_train_predict=model.predict(X_train)
y_predict=model.predict(X_test)
pd.crosstab(y_test,y_predict)

```

	col_0	3.0	4.0	5.0
clean_Priority				
	3.0	1059	487	89
	4.0	242	6708	222
	5.0	144	670	4361

Figure 6: Confusion matrix using Random Forest classifier

```
print(classification_report(y_test,y_predict))
```

	precision	recall	f1-score	support
3.0	0.73	0.65	0.69	1635
4.0	0.85	0.94	0.89	7172
5.0	0.93	0.84	0.89	5175
accuracy			0.87	13982
macro avg	0.84	0.81	0.82	13982
weighted avg	0.87	0.87	0.87	13982

Figure 7: Classification report using Random Forest classifier

8) PCA

```
pca=PCA()
X=pd.DataFrame(pca.fit_transform(X))
X.head()
pca.explained_variance_
pca.explained_variance_ratio_
Follow the same procedure like define x and y, label encoder, train-test split, as in Random-Forest for PCA,
XGBoost, ANN, KNN, Logistic regression
model=RandomForestClassifier(n_estimators=250,random_state=10,criterion='gini')
model.fit(X_train,y_train)
print(model.feature_importances_)
pd.DataFrame(model.feature_importances_,index=X.columns).sort_values(0,ascending=False)
pd.crosstab(y_test,y_predict)
```

```
:
```

col_0	3.0	4.0	5.0
clean_Priority			
3.0	1059	487	89
4.0	242	6708	222
5.0	144	670	4361

Figure 8: Confusion matrix using PCA

```
print(classification_report(y_test,y_predict))
```

	precision	recall	f1-score	support
3.0	0.73	0.65	0.69	1635
4.0	0.85	0.94	0.89	7172
5.0	0.93	0.84	0.89	5175
accuracy			0.87	13982
macro avg	0.84	0.81	0.82	13982
weighted avg	0.87	0.87	0.87	13982

Figure 9: Classification report using PCA

9) XGBoost

```

from xgboost import XGBClassifier
model=XGBClassifier(learning_rate=0.5,random_state=5,n_estimators=50)
model.fit(X_train,y_train)
y_train_predict=model.predict(X_train)
y_predict=model.predict(X_test)
pd.crosstab(y_test,y_predict)

```

338]:

col_0	3.0	4.0	5.0
clean_Priority			
3.0	1008	529	162
4.0	212	6606	288
5.0	125	758	4294

Figure 10: Confusion matrix using XGBoost

```
print(classification_report(y_test,y_predict))
```

	precision	recall	f1-score	support
3.0	0.75	0.59	0.66	1699
4.0	0.84	0.93	0.88	7106
5.0	0.91	0.83	0.87	5177
accuracy			0.85	13982
macro avg	0.83	0.78	0.80	13982
weighted avg	0.85	0.85	0.85	13982

Figure 11: Classification report using XGBoost

10) ANN

```

from sklearn.neural_network import MLPClassifier
Standardize the dataset along X-axis
X=scale(X)
model=MLPClassifier(hidden_layer_sizes=(55,67,50),random_state=10)
model.fit(X_train,y_train)
y_train_predict=model.predict(X_train)
y_predict=model.predict(X_test)
pd.crosstab(y_test,y_predict)

```

col_0	3.0	4.0	5.0
clean_Priority			
3.0	927	570	202
4.0	368	6179	559
5.0	180	955	4042

Figure 12: Confusion matrix using ANN

```
print(classification_report(y_test,y_predict))
```

	precision	recall	f1-score	support
3.0	0.63	0.55	0.58	1699
4.0	0.80	0.87	0.83	7106
5.0	0.84	0.78	0.81	5177
accuracy			0.80	13982
macro avg	0.76	0.73	0.74	13982
weighted avg	0.80	0.80	0.79	13982

Figure 13: Classification report using ANN

11) KNN

```

from sklearn.neighbors import KNeighborsClassifier
model= KNeighborsClassifier(n_neighbors=10)
model.fit(X_train,y_train)
y_train_predict=model.predict(X_train)
y_predict=model.predict(X_test)
pd.crosstab(y_test,y_predict)

```

	col_0	3.0	4.0	5.0
clean_Priority				
3.0	695	769	197	
4.0	353	6334	505	
5.0	180	1393	3556	

Figure 14: Confusion matrix using KNN

```
print(classification_report(y_test,y_predict))
```

	precision	recall	f1-score	support
3.0	0.57	0.42	0.48	1661
4.0	0.75	0.88	0.81	7192
5.0	0.84	0.69	0.76	5129
accuracy			0.76	13982
macro avg	0.72	0.66	0.68	13982
weighted avg	0.76	0.76	0.75	13982

Figure 15: Classification report using KNN**12) Logistic regression**

```

model=LogisticRegression()
model.fit(X_train,y_train)
y_train_predict=model.predict(X_train)
y_predict=model.predict(X_test)
pd.crosstab(y_test,y_predict)

```

	col_0	3.0	4.0	5.0
clean_Priority				
3.0	56	1367	63	
4.0	69	5699	196	
5.0	39	2161	2002	

Figure 16: Confusion matrix using Logistic Regression

```
print(classification_report(y_test,y_predict))
```

	precision	recall	f1-score	support
3.0	0.34	0.04	0.07	1486
4.0	0.62	0.96	0.75	5964
5.0	0.89	0.48	0.62	4202
accuracy			0.67	11652
macro avg	0.61	0.49	0.48	11652
weighted avg	0.68	0.67	0.62	11652

Figure 17: Classification report using Logistic Regression

2. Forecast the incident volume in different fields, quarterly and annual. So that they can be better prepared with resources and technology planning.

- 1) Check for EDA

```
Counter(data.Priority)
```

```
data.isnull().sum().to_frame().T
```

- 2) Create a new Dataframe containing Open_Time as parameter

```
data_new=data.groupby(data.Open_Time).sum()
```

```
data_new
```

```
data_new.info()
```

```
data_new.head().T
```

```
data_new=data.loc[:,['Open_Time','Incident_ID','CI_Name','CI_Cat','CI_Subcat','WBS','Category','Priority']]
```

```
data_new
```

```
data_new.dropna(inplace=True)
```

```
data_new.isna().sum().to_frame().T
```

```
data_new['Open_Date']=data_new['Open_Time'].apply(lambda x:x.date())
```

```
data_new.Open_Date
```

```

data_new['No_Incidents']=data_new.groupby('Open_Date')['Incident_ID'].transform('count')
data_new.head()

data_new.corr()
incidents=data_new.loc[:,['Open_Date','No_Incidents']]
incidents.head()

incidents.drop_duplicates(inplace=True)
incidents.shape
incidents.head()

incidents=incidents.set_index('Open_Date')
incidents.index=pd.to_datetime(incidents.index)
incidents.index

print(incidents.index.min(),to,incidents.index.max())
data=incidents['No_Incidents']
data=incidents.asfreq('D')
data.index

incidents.head()

data.plot(figsize=(16,7),marker='o')
plt.legend(loc='upper center',frameon=True, labelspace=1,fontsize=20)
plt.show()

```

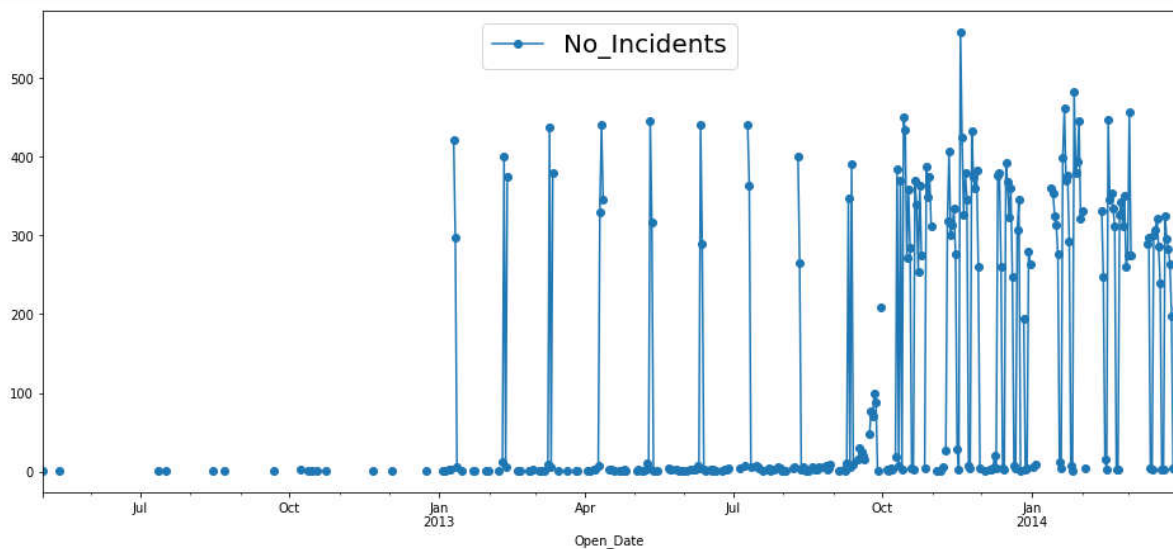


Figure 18: Line plot for Open_Date v/s No_of_Incidents

```

infrom2013=incidents[incidents.index>dt.datetime(2013,10,1)]
infrom2013
infrom2014=incidents[incidents.index>dt.datetime(2014,10,1)]
infrom2014

data=infrom2013['No_Incidents']
data.plot(figsize=(16,7),marker='o')

```



```
plt.legend(loc='upper center',frameon=True, labelspace=1,fontsize=20)
plt.show()
```

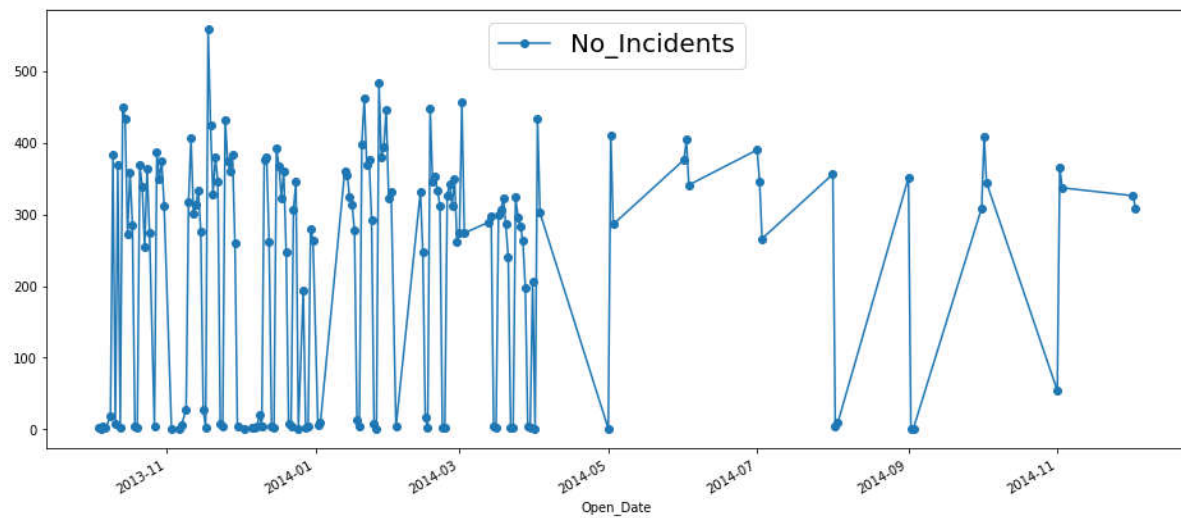


Figure 19: Line plot for Open_Date v/s No_of_Incidents from 2013

```
data=infrom2014['No_Incidents']
data.plot(figsize=(16,7),marker='o')
plt.legend(loc='upper center',frameon=True, labelspace=1,fontsize=20)
plt.show()
```

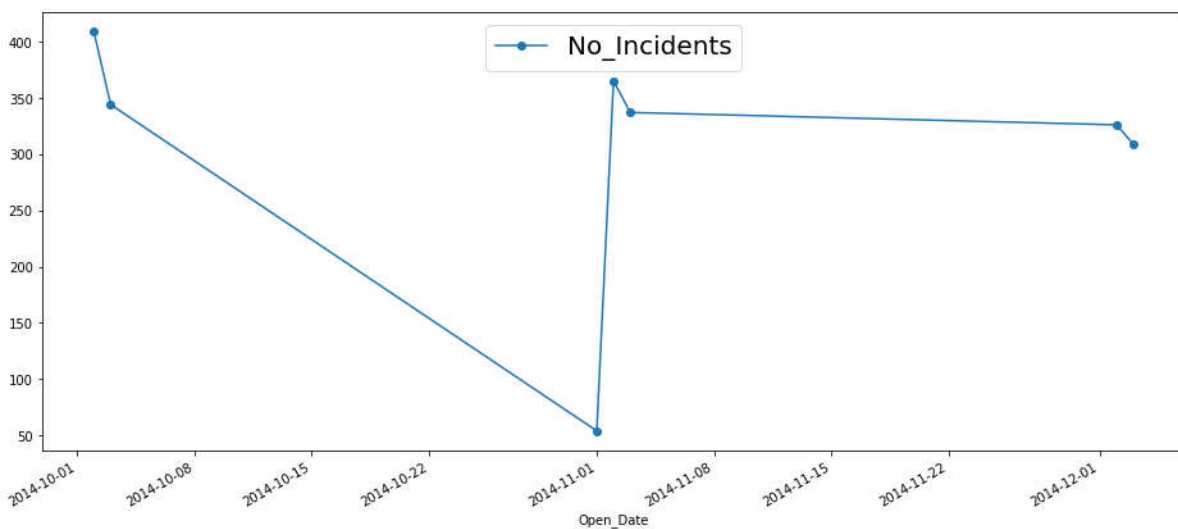


Figure 20: Line plot for Open_Date v/s No_of_Incidents from 2014

```
data=data.asfreq('D')
data.index
```

3) Time series Forecasting

```
incidents.plot(figsize=(21,10),marker='o')
```

```
plt.legend(loc='upper center',frameon=True, labelspace=1,fontsize=20)
plt.xlabel(xlabel='Resolved_Time',fontdict={'fontsize':20})
```

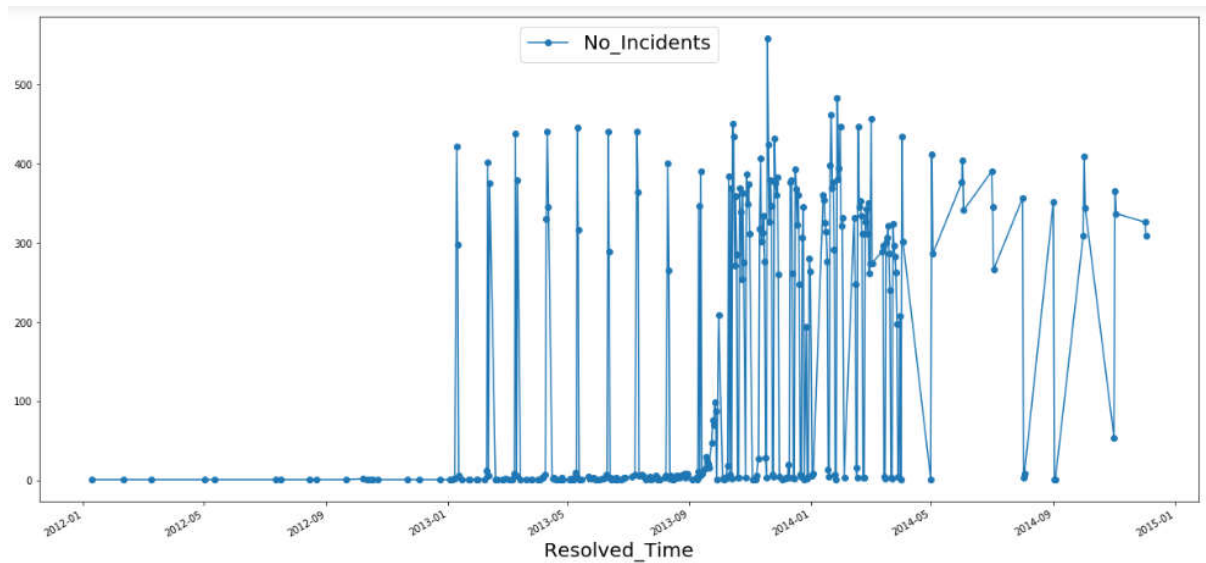
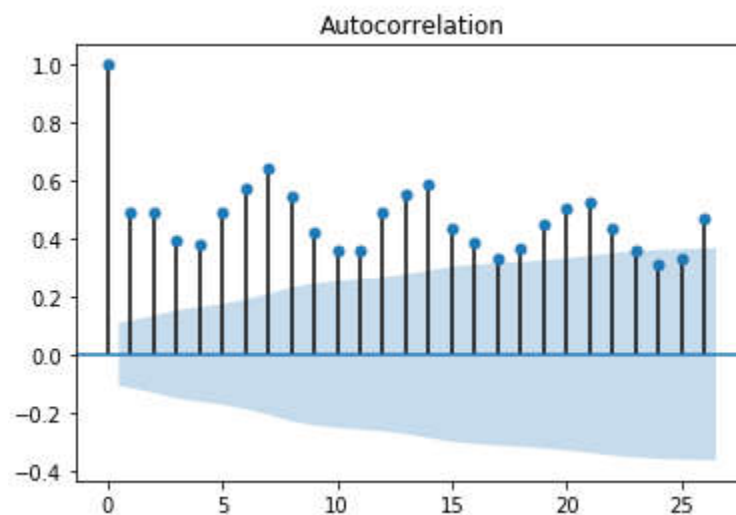


Figure 21: Time series forecasting line plot for Resolved_Time v/s No_of_Incidents

4) ACF and PACF

```
from statsmodels.graphics.tsaplots import plot_acf
plot_acf(incidents)
```



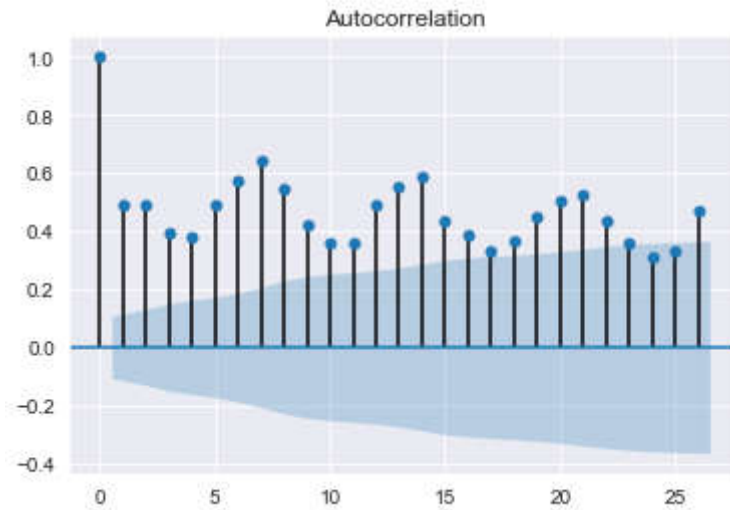


Figure 22: ACF plot for incidents

```
from statsmodels.graphics.tsaplots import plot_pacf
plot_pacf(incidents)
```

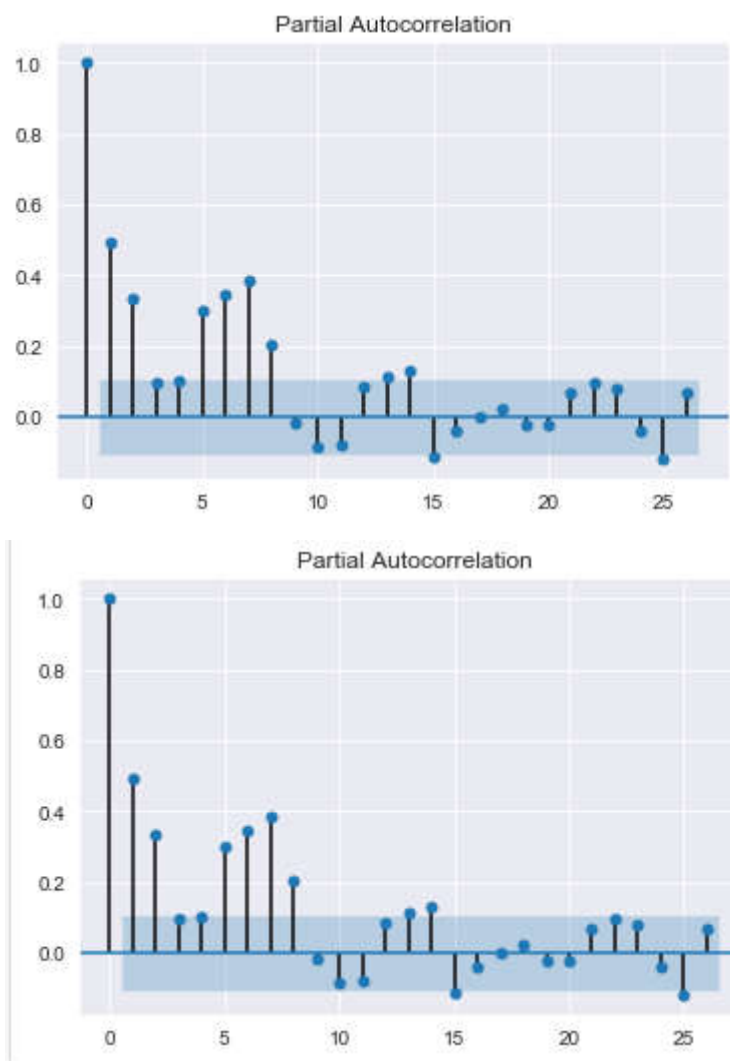


Figure 23: PACF plot for incidents

5) Converting to stationary

```
incidents_diff=incidents.diff(periods=1)
incidents_diff=incidents_diff[1:]
incidents_diff.head()
plot_acf(incidents_diff)
```

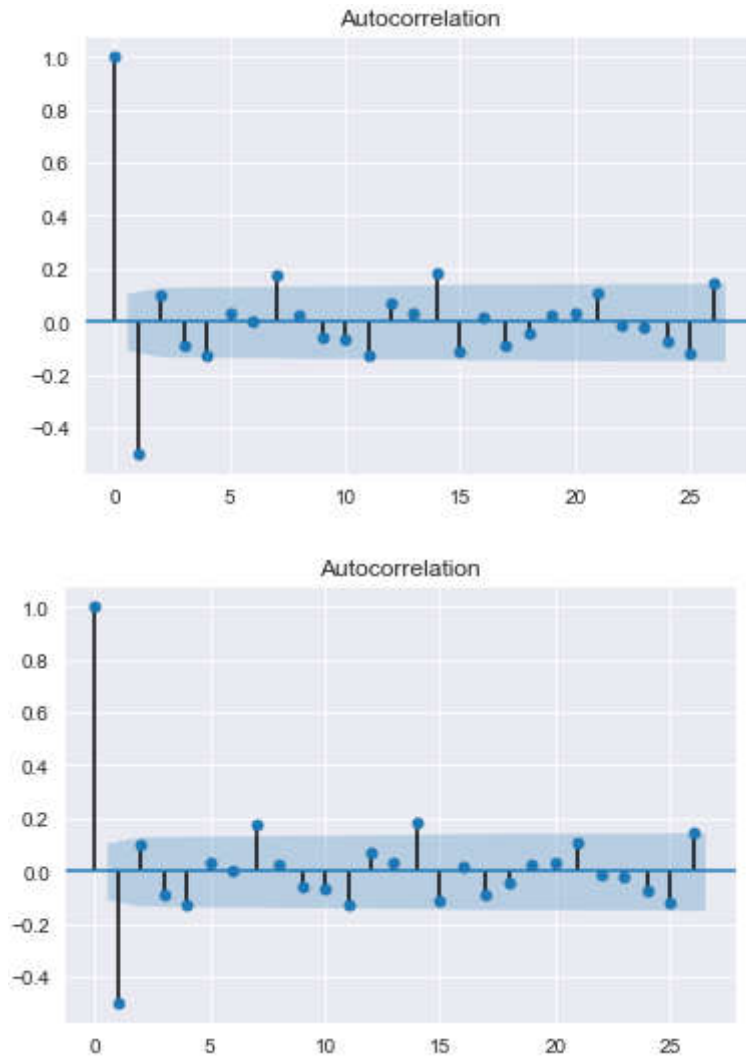


Figure 24: ACF plot for incidents after converting data to stationary

```
plot_pacf(incidents_diff)
```

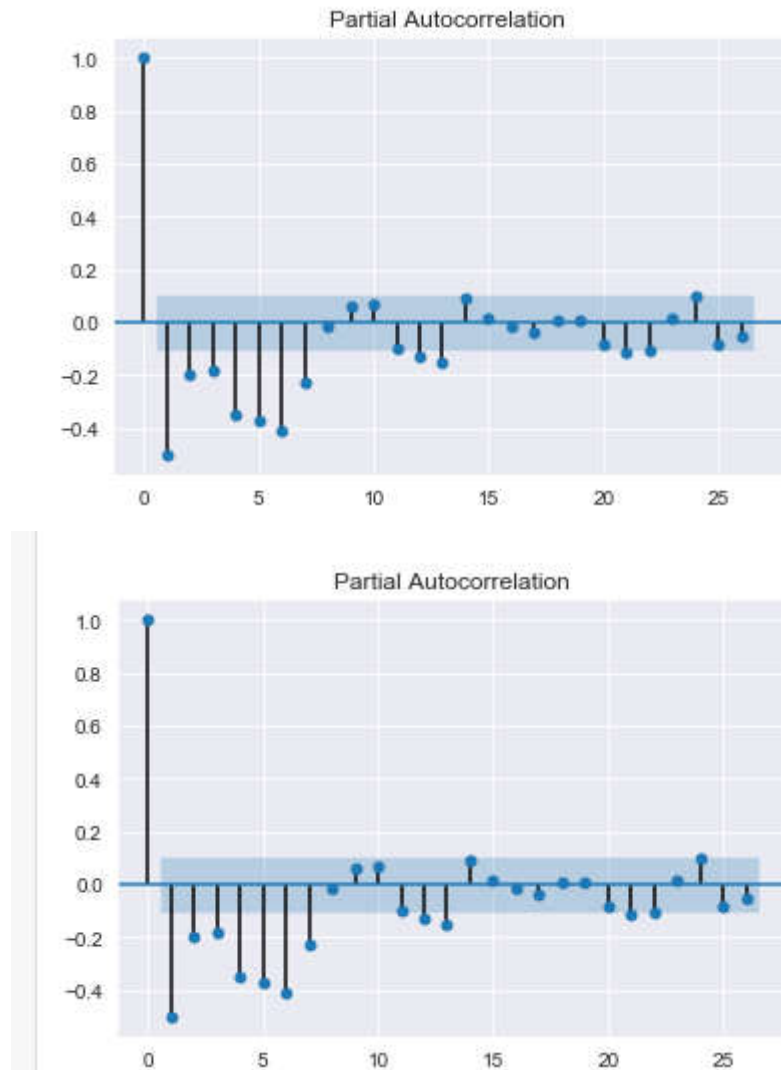


Figure 25: PACF plot for incidents after converting data to stationary

```
incidents_diff.plot(figsize=(21,10),marker='o')
sns.set_style("darkgrid")
plt.legend(loc='upper center',frameon=True, labelspace=1,fontsize=20)
plt.xlabel(xlabel='Resolved Time',fontdict={'fontsize':20})
```

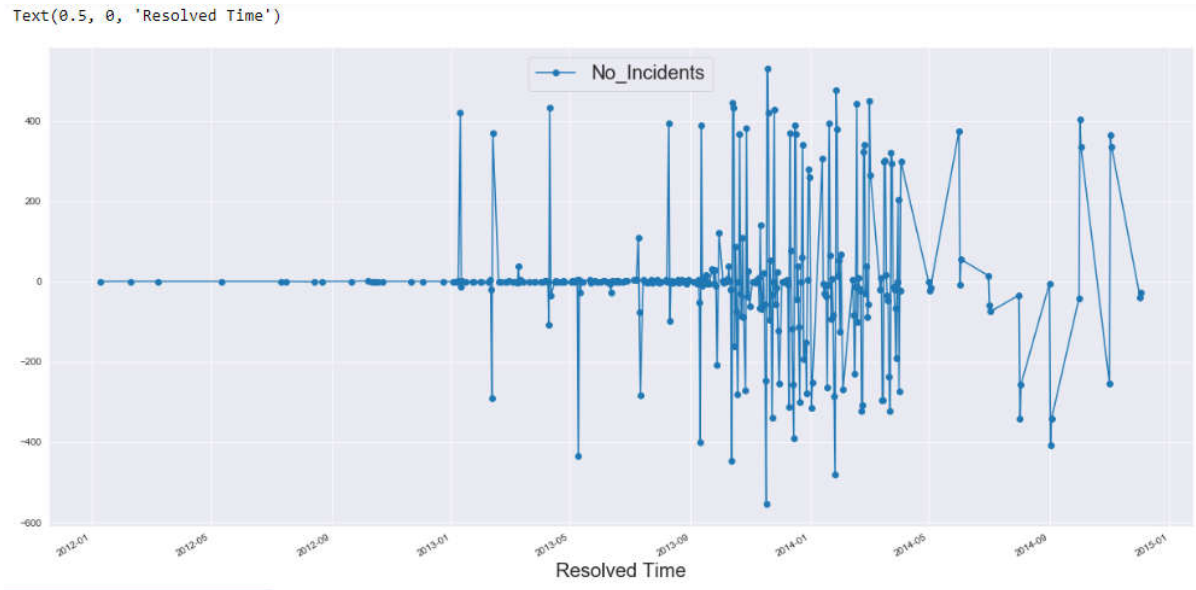


Figure 26: Time series forecasting line plot for Resolved_Time v/s No_of_Incidents for stationary data

```
incidents_diff.shape
incidents_diff.head()
incidents_diff.describe()
incidents_diff.info()
incidents_diff.isna().sum().to_frame().T
```

6) Auto Regressive Integrated Moving Average(ARIMA) Model

```
X=incidents_diff.values

train=X[:300]

test=X[301:]

predictions=[]

endog=train

from statsmodels.tsa.arima_model import ARIMA
from sklearn.metrics import mean_squared_error
model_arima=ARIMA(train,order=(10,1,1))
model_arima_fit=model_arima.fit()
print(model_arima_fit.aic)
predictions = model_arima_fit.forecast(steps=24,alpha=0.05)[0]
predictions

plt.figure(figsize=(16,5))
sns.set_style("darkgrid")
plt.plot(test,color='blue',label='Actual data',marker='o')
plt.plot(predictions,color='red',label="Predicted data",marker='o')
plt.legend(loc='upper center',frameon=True,fontsize=15)
```

```
plt.title('Prediction using ARIMA Model',fontdict={'fontsize':25},color='green')
plt.show()
```

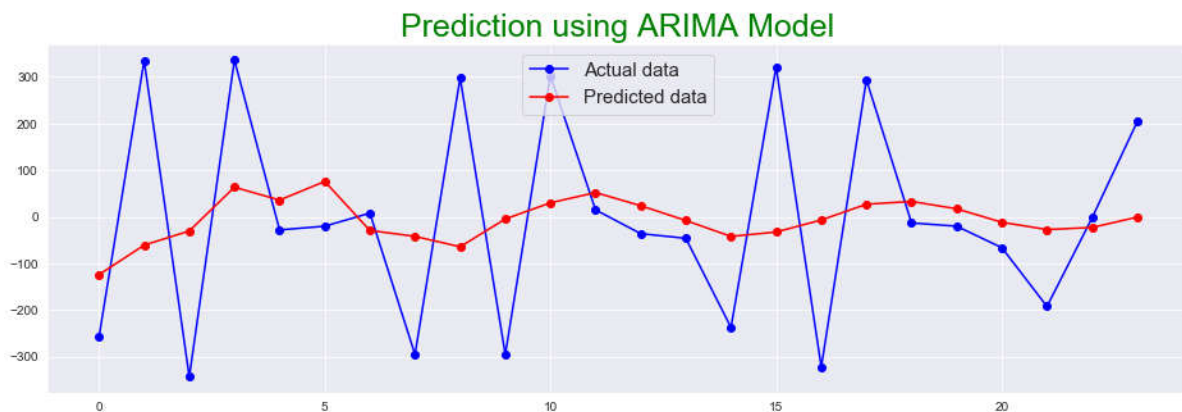


Figure 27: Prediction using ARIMA model

```
mean_squared_error(test,predictions)
np.sqrt(mean_squared_error(test,predictions))
import itertools
p=d=q=range(0,2)
pdq=list(itertools.product(p,d,q))
pdq

import warnings
warnings.filterwarnings('ignore')
for param in pdq:
    try:
        model_arima=ARIMA(train,order=param)
        model_arima_fit=model_arima.fit()
        print(param,model_arima_fit.aic)
    except:
        continue
```

7) Rolling Forecast on ARIMA

Using ARIMA model, the plot for predicted data was deviating from the actual data at many points. So I had used Rolling forecast technique to correct the graph for the predicted data.

A rolling window model involves calculating a statistic on a fixed contiguous block of prior observations and using it as a forecast.

It is much like the expanding window, but the window size remains fixed and counts backwards from the most recent observation.

It may be more useful on time series problems where recent lag values are more predictive than older lag values.

```
history = [x for x in train]
```

```
predictions = list()
```

```
for t in range(len(test)):
```

```

model = ARIMA(history, order=(4,2,0))

model_fit = model.fit()

output = model_fit.forecast()

y_predict= output[0]

predictions.append(y_predict)

observation = test[t]

history.append(observation)

print('predicted=%f, expected=%f' % (y_predict, observation))

error = mean_squared_error(test, predictions)

print("Test MSE: %.3f % error)

print('Square root of Test MSE: %.3f %np.sqrt(error))

plt.figure(figsize=(16,7))

plt.plot(test,color='blue',label='Actual data',marker='o')

plt.plot(predictions, color='red',label='Predicted data',marker='o')

plt.legend(loc='upper right',frameon=True,fontsize=15)

plt.title('Rolling forecast on ARIMA Model',fontdict={'fontsize':25},color='black')

plt.show()

```

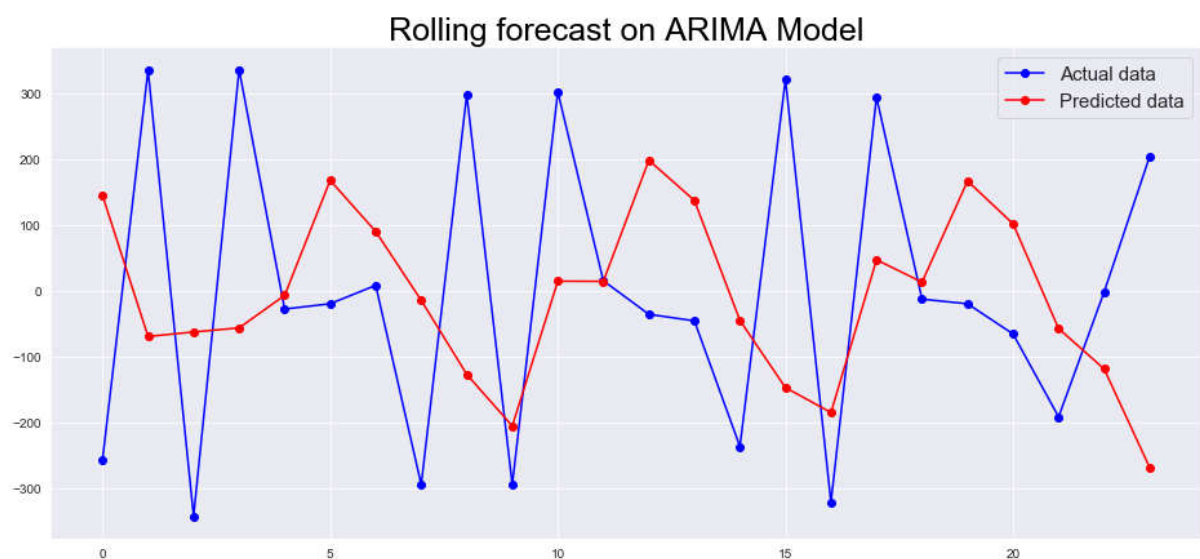


Figure 28: Rolling Forecast on ARIMA model

8) Seasonal Auto Regressive Integrated Moving Average(SARIMA) Model


```

from statsmodels.tsa.statespace.sarimax import SARIMAX

from sklearn.metrics import mean_squared_error

model_sarimax = SARIMAX(train,order=(2,2,1), seasonal_order=(2,2,1,2),mle_regression=True,
                        enforce_stationarity=True,enforce_invertibility=True)

print(model_sarimax)

model_sarimax_fit = model_sarimax.fit()

print(model_sarimax_fit.aic)plt.figure(figsize=(16,5))

sns.set_style("darkgrid")

plt.plot(test,color='blue',label='Actual data',marker='o')

plt.plot(predictions,color='red',label="Predicted data",marker='o')

plt.legend(loc='upper right',frameon=True,fontsize=15)

plt.title('Prediction using SARIMA Model',fontdict={'fontsize':25},color='green')

plt.show()

```

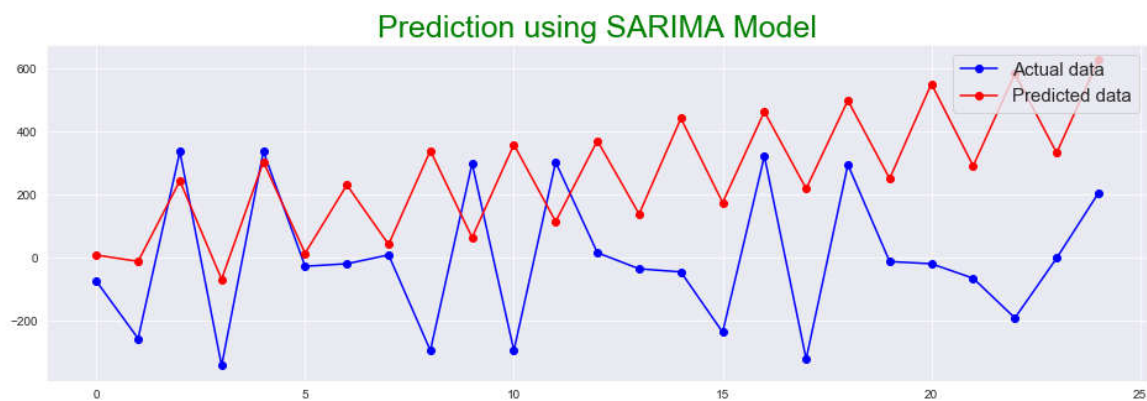


Figure 29: Prediction using SARIMA model

```

mean_squared_error(test,predictions)
np.sqrt(mean_squared_error(test,predictions))

import itertools

p=d=q=s=range(0,2)

pdqs=list(itertools.product(p,d,q,s))

pdqs

```

```

import warnings

warnings.filterwarnings('ignore')

for a in pdqs:

    try:

        model_sarimax=SARIMAX(train,order=a,seasonal_order=a)

        model_sarimax_fit=model_sarimax.fit()

        print(a,model_sarimax_fit.aic)

        print(model_fit.summary())

```

```

except:

```

```

    continue

```

Rolling forecast on SARIMA

```

history = [x for x in train]

predictions = list()

for t in range(len(test)):

    model = SARIMAX(history, order=(2,1,1),seasonal_order=(1,1,1,1))

    model_fit = model.fit()

    output = model_fit.forecast()

    y_predict= output[0]

    predictions.append(y_predict)

    observation = test[t]

    history.append(observation)

    print('predicted=%f, expected=%f' % (y_predict, observation))

```

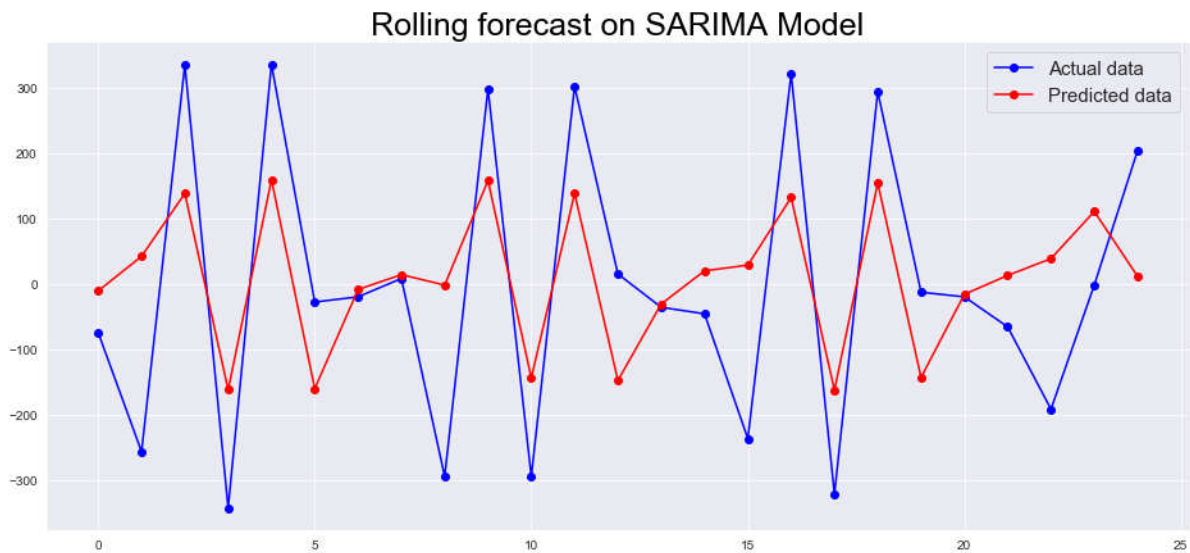


Figure 30: Rolling Forecast on SARIMA model

3. Auto tag the tickets with right priorities and right departments so that reassigning and related delay can be reduced.

1) Checking for outliers

```
sns.set_style('whitegrid')
sns.heatmap(data.isnull(),yticklabels=False,cbar=True,cmap='viridis')
```

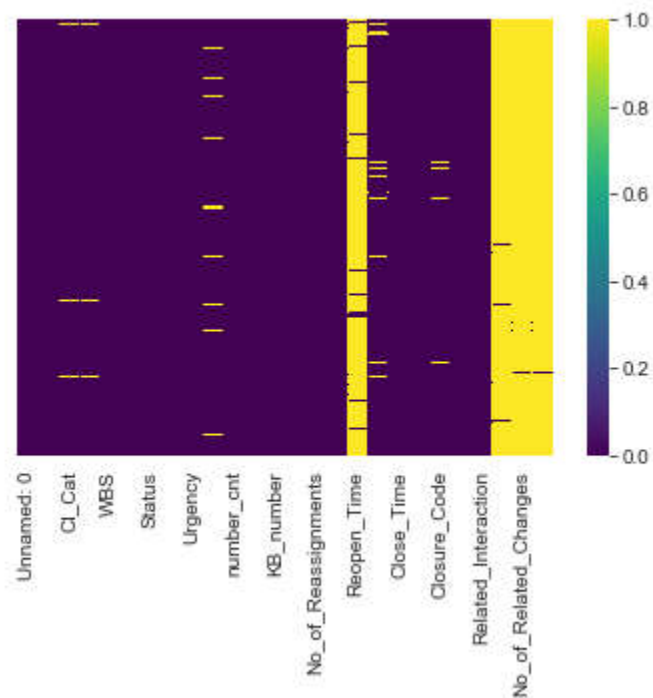


Figure 31: Heatmap for detecting outliers

```
data[['No_of_Reassignments']].boxplot();
```

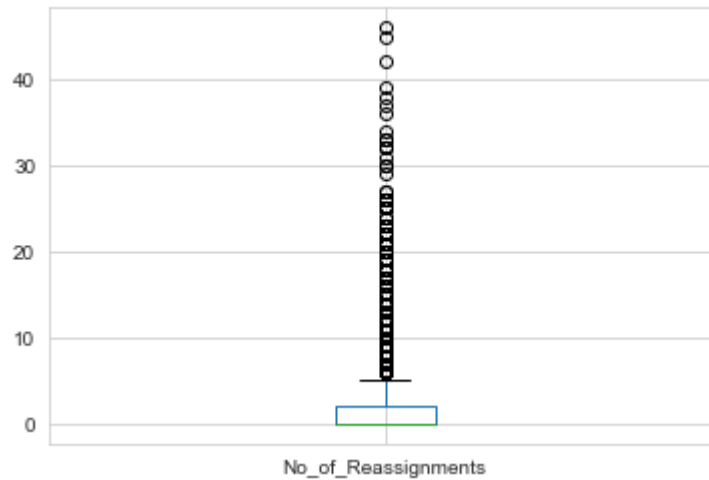


Figure 32: Boxplot to detect outliers

```
X=data.No_of_Reassignments
removed_outliers_Reassignments=X.between(X.quantile(0.7),X.quantile(0.9))
print(str(X[removed_outliers_Reassignments].size)+"/"+str(X.size)+" data points remain")
plt.boxplot(X[removed_outliers_Reassignments]);
plt.xlabel("No_of_Reassignments")
Text(0.5, 0, 'No_of_Reassignments')
```

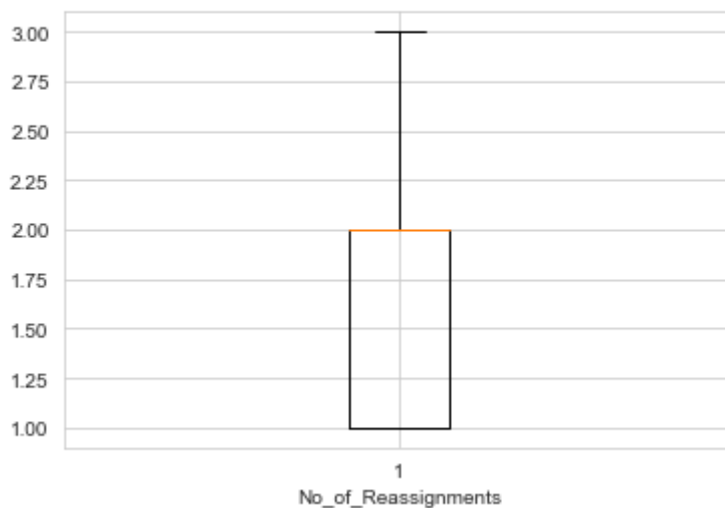


Figure 33: Boxplot to remove outliers

```
figure,axis=plt.subplots(1,2,figsize=(16,5))
axis[0].boxplot(X);
axis[1].boxplot(X[removed_outliers_Reassignments]);
axis[0].set_title("With outliers")
axis[0].set_xlabel("No_of_Reassignments")
axis[1].set_title("Removed outliers")
axis[1].set_xlabel("No_of_Reassignments")
```



Figure 33: Boxplot representing data with outliers and after removing outliers

```
data['clean_Reassignments']=X[removed_outliers_Reassignments]
```

2) Check for EDA

```
data=data.drop(['Urgency','Impact','Alert_Status','Open_Time','Reopen_Time','Close_Time','Resolved_
Time','No_of_Reassignments'],axis=1)
Counter(data.clean_Reassignments).most_common()
data.head()
dataset=data.loc[:,['CI_Cat','CI_Subcat','WBS','Category']]
dataset.head()
dataset.shape
dataset.dropna(inplace=True)
dataset.isna().sum().to_frame().T
dataset.info()
dataset.describe()
Create a new field tickets
dataset.loc[data.clean_Reassignments>4.0,'tickets']='high'
dataset.loc[(data.clean_Reassignments>2.0)& (data.clean_Reassignments<=4.0),'tickets']='medium'
dataset.loc[data.clean_Reassignments<=2.0,'tickets']='low'
dataset.head(34)
```

3) Define X and y

```
X=dataset.loc[:,dataset.columns!='tickets']
y=dataset.tickets
X.head()
y.fillna(method='ffill',inplace=True)
y.fillna(method='bfill',inplace=True)
y.isna().sum()
```

4) Using Label Encoder

```
enc=LabelEncoder()
```

```
X.CI_Cat=enc.fit_transform(X.CI_Cat)
X.CI_Subcat=enc.fit_transform(X.CI_Subcat)
X.WBS=enc.fit_transform(X.WBS)
X.Category=enc.fit_transform(X.Category)
```

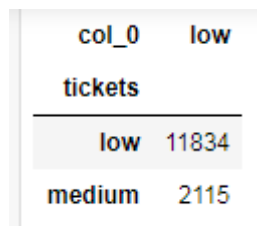
```
X.head()
X.info()
```

4) Using train-test split

```
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=20,test_size=0.3)
print("X_train shape = ",X_train.shape)
print("X_test shape = ",X_test.shape)
print("y_train shape = ",y_train.shape)
print("y_test shape = ",y_test.shape)
```

5) Random-forest classifier

```
model=RandomForestClassifier(n_estimators=300,random_state=10,max_depth=4,criterion='gini')
model.fit(X_train,y_train)
y_train_predict=model.predict(X_train)
y_predict=model.predict(X_test)
pd.crosstab(y_test,y_predict)
```



	col_0	low
tickets		
low	11834	
medium	2115	

Figure 34: Confusion matrix using Random-Forest Classifier

```
print(classification_report(y_test,y_predict))
```

	precision	recall	f1-score	support
low	0.85	1.00	0.92	11834
medium	0.00	0.00	0.00	2115
accuracy			0.85	13949
macro avg	0.42	0.50	0.46	13949
weighted avg	0.72	0.85	0.78	13949

Figure 35: Classification report using Random-Forest Classifier

6) XGBoost

```
model=XGBClassifier(max_depth=3,learning_rate=0.5,random_state=50,n_estimators=50)
model.fit(X_train,y_train)
y_train_predict=model.predict(X_train)
y_predict=model.predict(X_test)
pd.crosstab(y_test,y_predict)
```

col_0	low
tickets	
low	11834
medium	2115

Figure 36: Confusion matrix using XGBoost

```
print(classification_report(y_test,y_predict))
```

	precision	recall	f1-score	support
low	0.85	1.00	0.92	11834
medium	0.00	0.00	0.00	2115
accuracy			0.85	13949
macro avg	0.42	0.50	0.46	13949
weighted avg	0.72	0.85	0.78	13949

Figure 37: Classification report using XGBoost

4. Predict RFC (Request for change) and possible failure / misconfiguration of ITSM assets.

1) Import the necessary packages

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib import rcParams
%matplotlib inline
from collections import Counter
import warnings
warnings.filterwarnings("ignore")
from sklearn.preprocessing import LabelEncoder,scale
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score,precision_score,confusion_matrix,classification_report,f1_score,recall_score
import datetime as dt
from scipy import stats
import pandas.util.testing as tm
```

2) Load the dataset

```
data_parser=lambda c: pd.to_Dataframe(c,format='%d/%m/%Y %H:%M:%s')
```

```
data=pd.read_csv('C:\\Users\\DELL\\Desktop\\Rubixe      projects\\Mar2020\\ITSM_data.csv',
parse_dates=['Open_Time','Reopen_Time','Close_Time','Resolved_Time'])
data.head()
data.shape
data.info()
data.describe()
```

3) Check for EDA

```
data=data.drop(['Urgency','Impact','Alert_Status','No_of_Related_Incidents','Status','Open_Ti
me','Reopen_Time','Close_Time','Resolved_Time'],axis=1)
data=data.iloc[:,:]
data.head()
data.shape
data.isna().sum().to_frame().T
data['No_of_Related_Changes'].fillna(method='ffill',inplace=True)
data['No_of_Related_Changes'].fillna(method='bfill',inplace=True)
data['Related_Change'].fillna(method='ffill',inplace=True)
data['Related_Change'].fillna(method='bfill',inplace=True)
data.isna().sum().to_frame().T
data.dropna(inplace=True,how='any')
data.info()
data.describe()
```

4) Define X and y

```
X=data.loc[:,data.columns!='No_of_Related_Changes']
y=data.No_of_Related_Changes
X.info()
```

5) Using Label Encoder

```
enc=LabelEncoder()
X.CI_Name=enc.fit_transform(X.CI_Name)
X.CI_Cat=enc.fit_transform(X.CI_Cat)
X.Incident_ID=enc.fit_transform(X.Incident_ID)
X.CI_Subcat=enc.fit_transform(X.CI_Subcat)
X.WBS=enc.fit_transform(X.WBS)
X.Category=enc.fit_transform(X.Category)
X.KB_number=enc.fit_transform(X.KB_number)
X.Related_Interaction=enc.fit_transform(X.Related_Interaction)
X.Handle_Time_hrs=enc.fit_transform(X.Handle_Time_hrs)
X.Closure_Code=enc.fit_transform(X.Closure_Code)
X.Related_Change=enc.fit_transform(X.Related_Change)
X.head()
X.info()
```

6) Using train-test split

```
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=20,test_size=0.3)
```



```
print("X_train shape = ",X_train.shape)
print("X_test shape = ",X_test.shape)
print("y_train shape = ",y_train.shape)
print("y_test shape = ",y_test.shape)
```

7) Random-Forest classifier

```
model=RandomForestClassifier(n_estimators=250,random_state=10,max_depth=4)
model.fit(X_train,y_train)
y_train_predict=model.predict(X_train)
y_predict=model.predict(X_test)
pd.crosstab(y_test,y_predict)
```

	col_0	1.0	2.0	3.0	9.0
No_of_Related_Changes					
1.0	12663	0	0	0	
2.0	0	672	0	0	
3.0	0	0	29	0	
9.0	0	0	0	3	

Figure 38: Confusion matrix using Random-Forest Classifier

```
print(classification_report(y_test,y_predict))
```

	precision	recall	f1-score	support
1.0	1.00	1.00	1.00	12663
2.0	0.97	1.00	0.99	672
3.0	1.00	0.48	0.65	29
9.0	0.00	0.00	0.00	3
accuracy			1.00	13367
macro avg	0.74	0.62	0.66	13367
weighted avg	1.00	1.00	1.00	13367

Figure 39: Classification report using Random-Forest Classifier

8) XGBoost

```
model=XGBClassifier(max_depth=3,learning_rate=0.5,random_state=50,n_estimators=50)
model.fit(X_train,y_train)
y_train_predict=model.predict(X_train)
y_predict=model.predict(X_test)
pd.crosstab(y_test,y_predict)
```

	col_0	1.0	2.0	3.0	9.0
No_of_Related_Changes					
1.0	12663	0	0	0	
2.0	0	672	0	0	
3.0	0	0	29	0	
9.0	0	0	0	3	

Figure 40: Confusion matrix using XGBoost

```
print(classification_report(y_test,y_predict))
```

	precision	recall	f1-score	support
1.0	1.00	1.00	1.00	12663
2.0	1.00	1.00	1.00	672
3.0	1.00	1.00	1.00	29
9.0	1.00	1.00	1.00	3
accuracy			1.00	13367
macro avg	1.00	1.00	1.00	13367
weighted avg	1.00	1.00	1.00	13367

Figure 41: Classification report using XGBoost