Data Science PROJECT

Client: NewX Services

Category: Forecasting Spare parts inventory

Project Ref: PM-PR-0027

TEAM MEMBERS:

1) **Ishan Pradip Borker**
2) **Kingston Teffy Roy**
3) **Ravichandra Reddy**
4) **Ayyappa Thalawar**
5) **Shilpa V.**

BUSINESS CASE:

The case business case is on the inventory management. Keeping Inventory of spare in various service centre to the market demand is always a challenge as most service centres spends significant amount in spare parts inventory costs. In spite of this, availability of spare parts is been one of the problem areas.

PROJECT GOAL:

Create Predictive model for inventory forecasting so that service centre achieve JIT standards.

FEATURE DETAILS:

Range Index: 28484 entries, 0 to 28483
Data columns (total 7 columns):
Invoice Date 28482 non-null datetime64[ns]
Job Card Date 28482 non-null datetime64[ns]
Business Partner Name 28482 non-null object
Vehicle No. 28484 non-null object
Vehicle Model 28482 non-null object
Current KM Reading 28482 non-null float64
INVOICE LINE TEXT 28449 non-null object

Assumptions

- Used two fields for the project. i.e. Job_Card_Date and INVOICE_LINE_TEXT.
- Created a new field named orders on demand from INVOICE_LINE_TEXT for using it in time series forecasting along with Job_Card_Date.
- Plotted graphs for finding the trends and getting the insights of the spare service parts dataset.
- Did not use Invoice_Date for prediction.
- Used more data in train dataset than the test dataset.

Approach

1. Import all the necessary packages like
    - **import pandas as pd**
    - **import numpy as np**
    - **from datetime import datetime**
    - **import matplotlib.pyplot as plt**
    - **from matplotlib import pyplot**
    - **from matplotlib import rcParams**
    - **%matplotlib inline**
    - **from scipy import stats**
    - **from collections import Counter**
    - **import warnings**
    - **warnings.filterwarnings('ignore')**

2. Define the function named parser.
   It will parse the data in yyyy-mm-dd format.
   ```
   def parser(x):
      return datetime.strptime(x,'%d-%m-%y').date()
   ```

3. Load the dataset.

   ```
   data=pd.read_excel("C:\\Users\DELL\Desktop\Rubixe projects\FEB2020\service-data.xlsx",header=0,date_parser='Job_Card_Date')
   data.head()
   ```

4. Describe the data

   ```
   data.describe()
   data.info()
   ```

5. Checking for the outliers

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

```
<matplotlib.axes._subplots.AxesSubplot at 0xbbf2388>
```

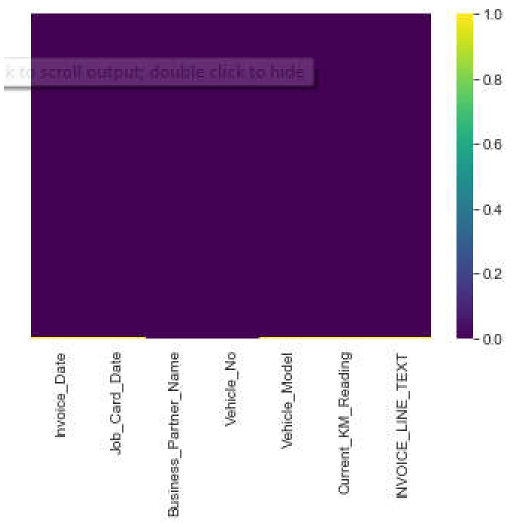

Fig 1: heatmap representation

6. Create Orders on demand column in the DataFrame

   data.loc[data.INVOICE_LINE_TEXT!=0,'Orders_on_Demand']=1
   data.head(4)

7. Find difference of days between job card date and invoice date.
   data['date_difference']=data['Invoice_Date']-data['Job_Card_Date']

   Use the counter to count the date difference between job card date and invoice date.
   Counter(data.date_difference)

   To check for the day difference of 19 days use the following code:
   data[data.date_difference =='19 days']

8. Check for the EDA Steps

   data.shape → For checking the shape of the data
   Counter(data.INVOICE_LINE_TEXT) → To count the number of each spare part
   data.isnull().sum().to_frame() → To check for the null values in the data
   data.dropna(axis=0,inplace=True) → To drop all the null values present in the data

9. Convert argument to datetime

   data['Job_Card_Date']=pd.to_datetime(data['Job_Card_Date'])
   type(data['Job_Card_Date'].iloc[0]) → Check the type
   data['Job_Card_Date'] → Display the details of Job Card Date

10. Split the data using DatetimeIndex function
    - **data['Year'] = pd.DatetimeIndex(data['Job_Card_Date']).year →**
    To separate year from the job card date
    - **data['Month'] = pd.DatetimeIndex(data['Job_Card_Date']).month →**
    To separate month from the job card dates

- **data.groupby('INVOICE_LINE_TEXT').count() → Count the value of each spare part and group them**

11. Finding the trends

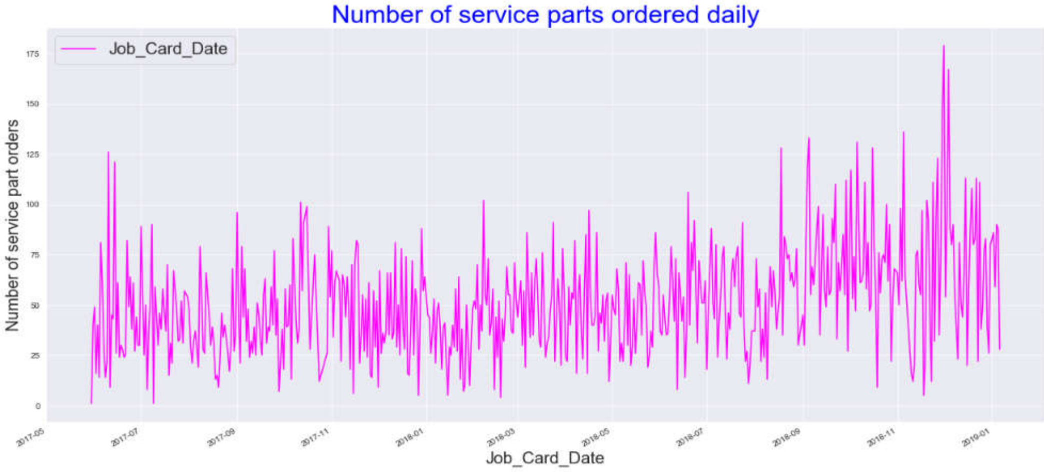

Fig 2: Line plot for number of service parts that are ordered on daily basis

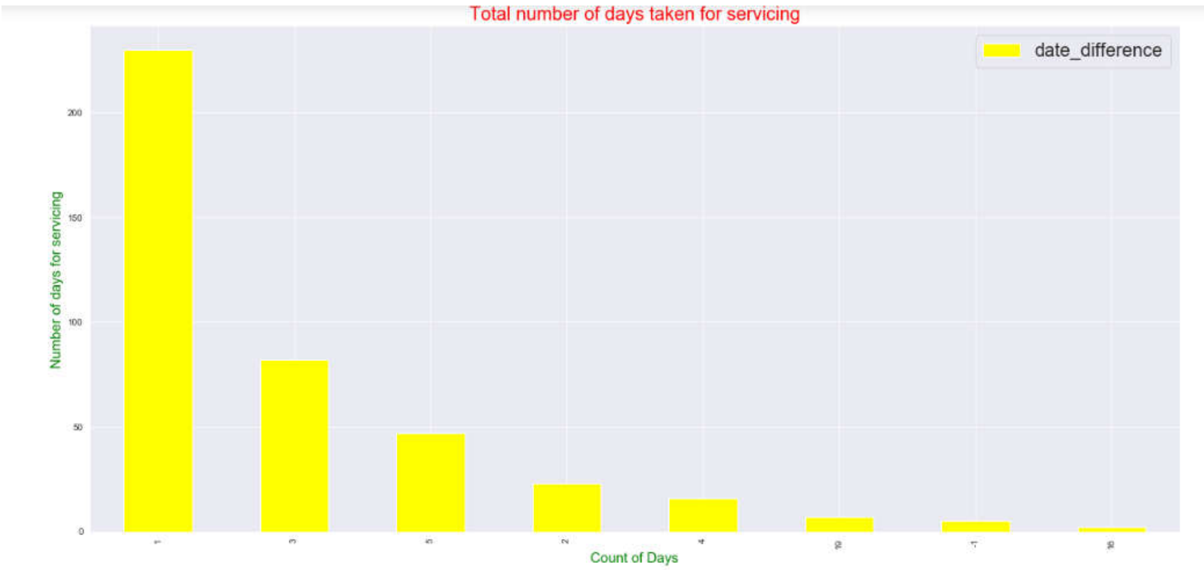

Fig 3: Bar plot representation for total number of days taken for servicing of a vehicle

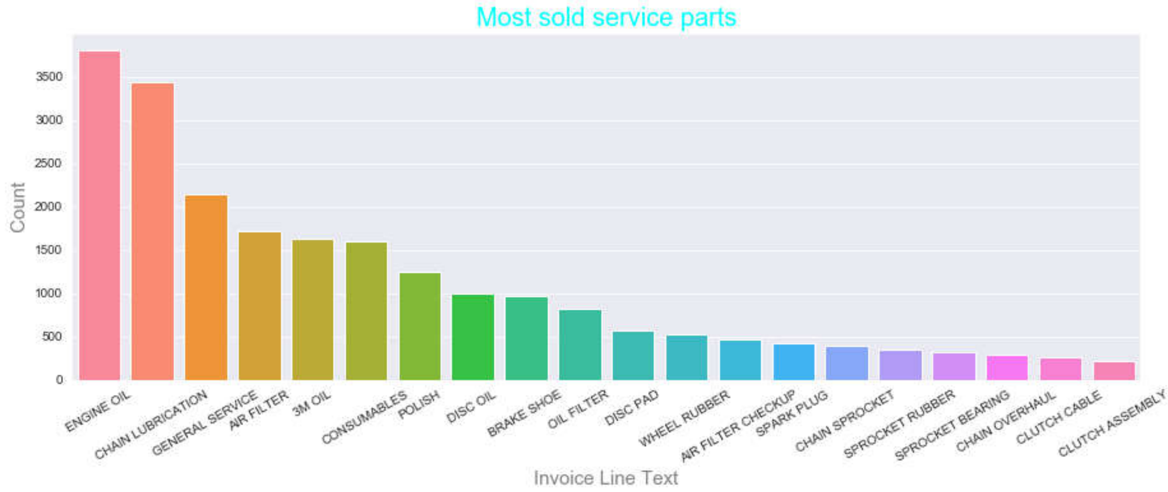

Fig 4: Bar plot representing most sold service parts

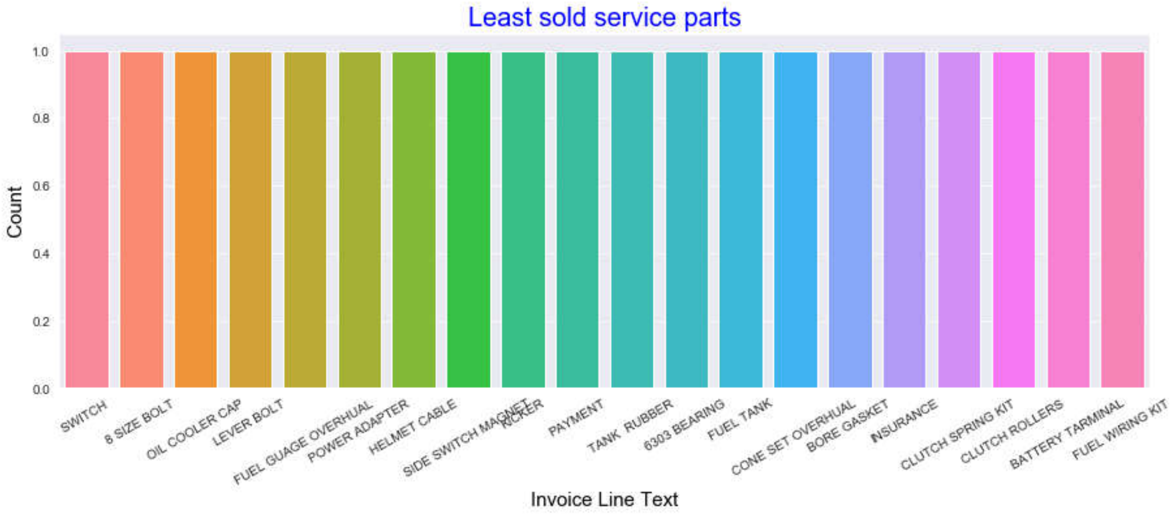

Fig 5: Bar plot representing least sold service parts

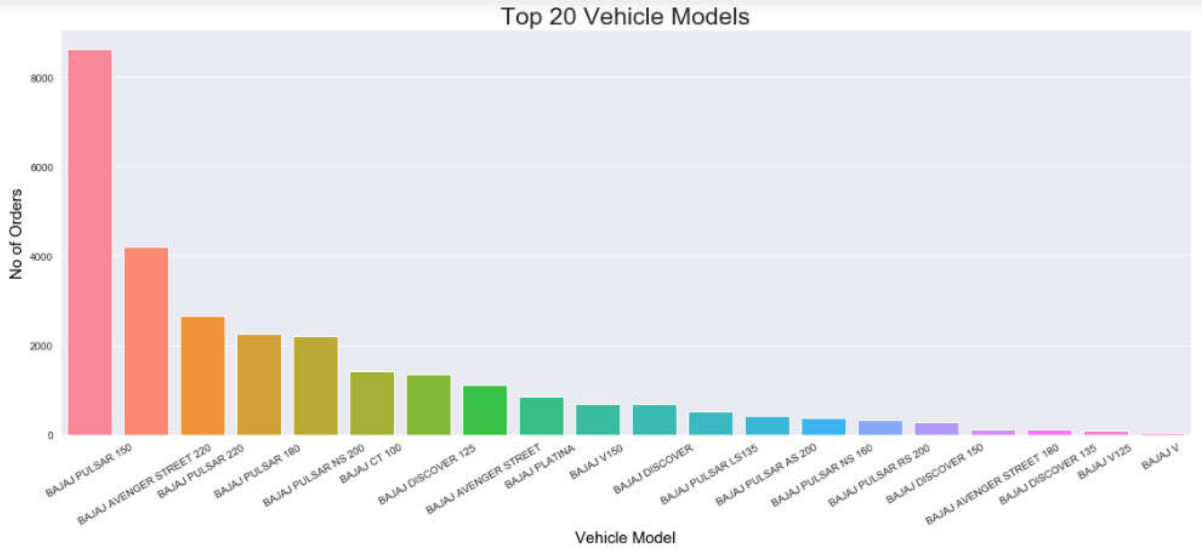

Fig 6: Bar plot representing Top 20 Vehicle Models

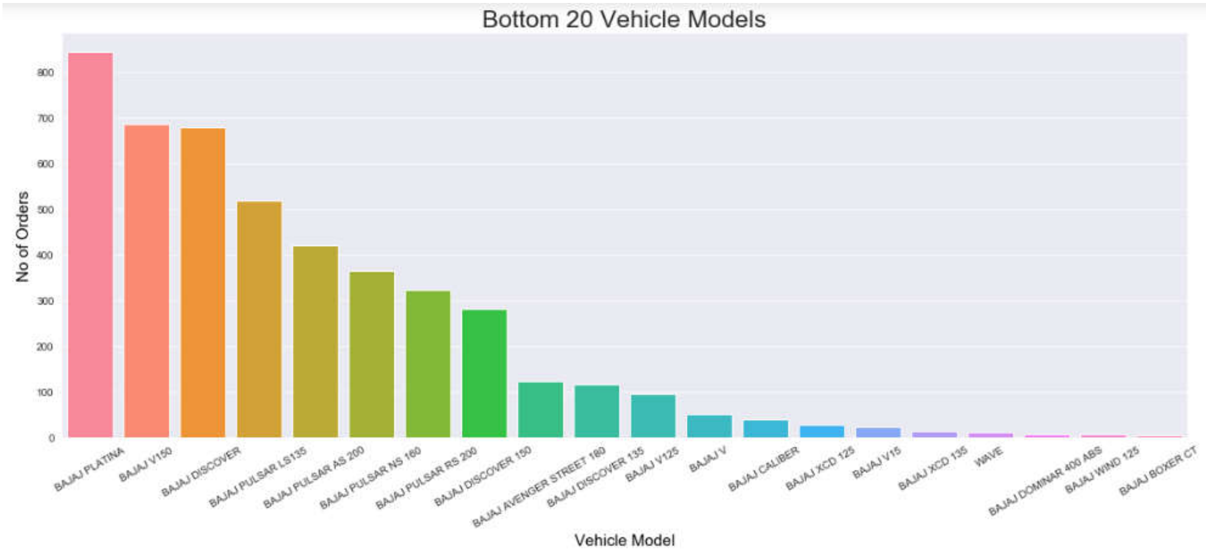

Fig 7: Bar plot representing Bottom 20 Vehicle Models

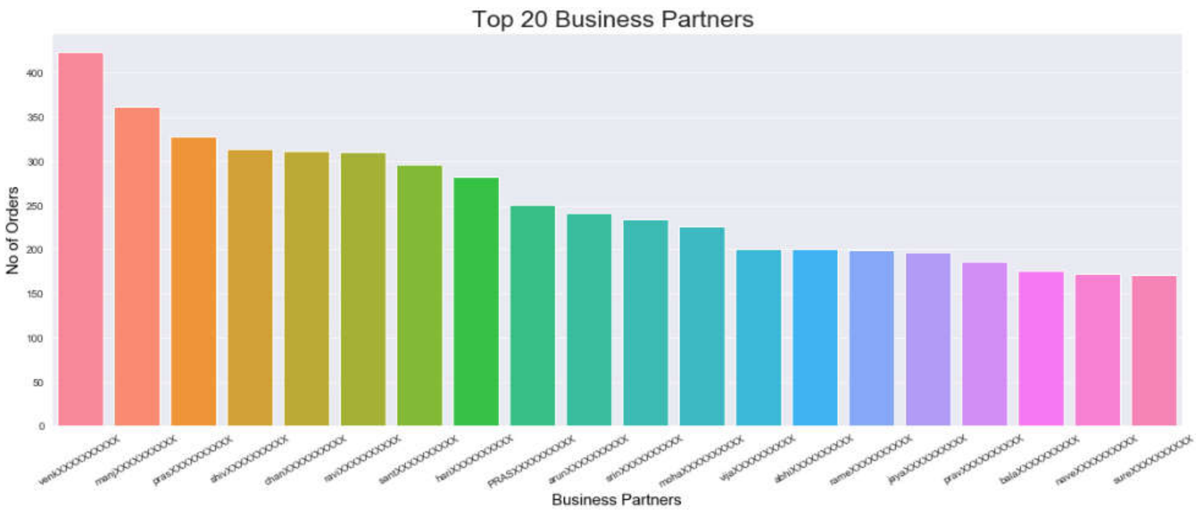

Fig 8: Bar plot representing Top 20 Business Partners

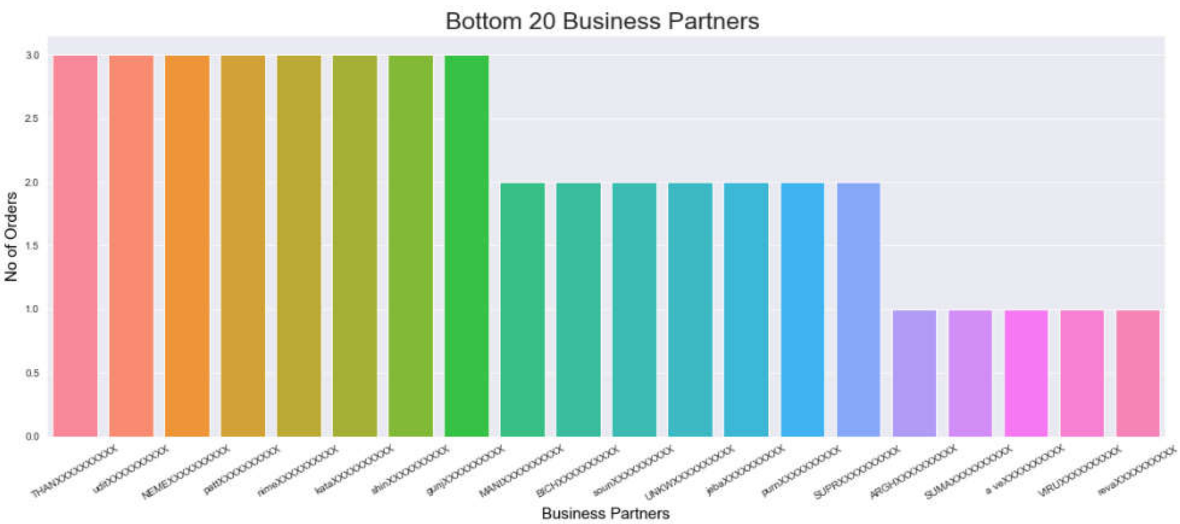

Fig 9: Bar plot representing Bottom 20 Business Partners

12. Creating a new Dataframe containing Invoice Line Text and Orders on Demand
    data_new=data.groupby(data.Job_Card_Date).sum()

13. Timeseries Forecasting

Fig 10: Line plot representing orders on demand with respect to Job Card Date for data_new dataframe

14. Plot the autocorrelation function(acf)

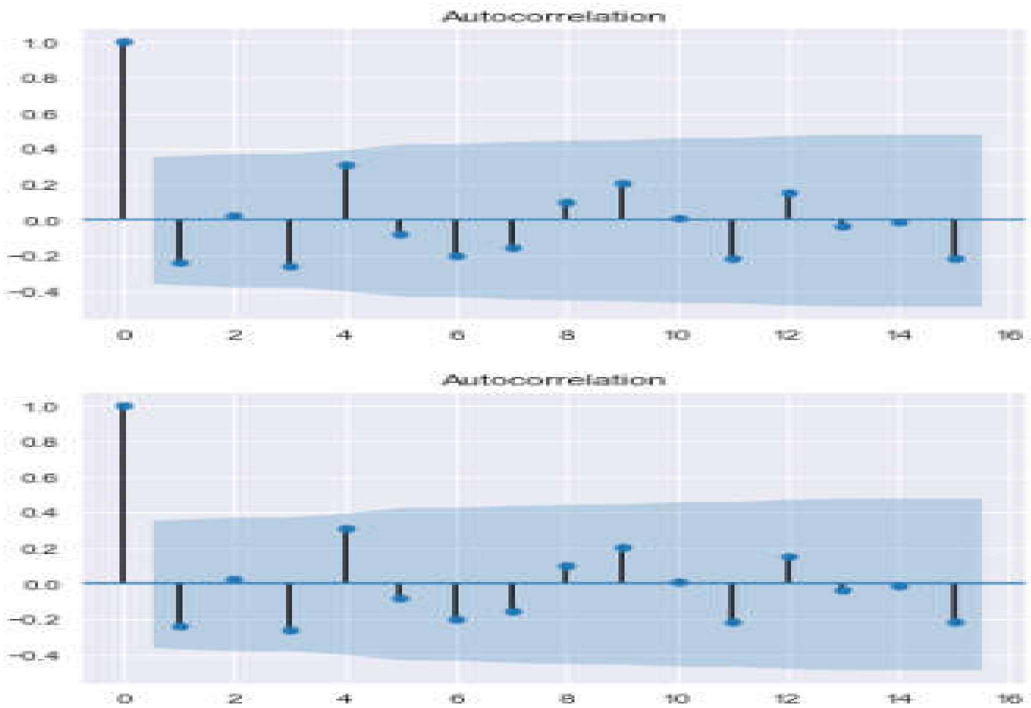

Fig 11: Auto correlation function (ACF)

15. Plot the partial autocorrelation function(pacf)

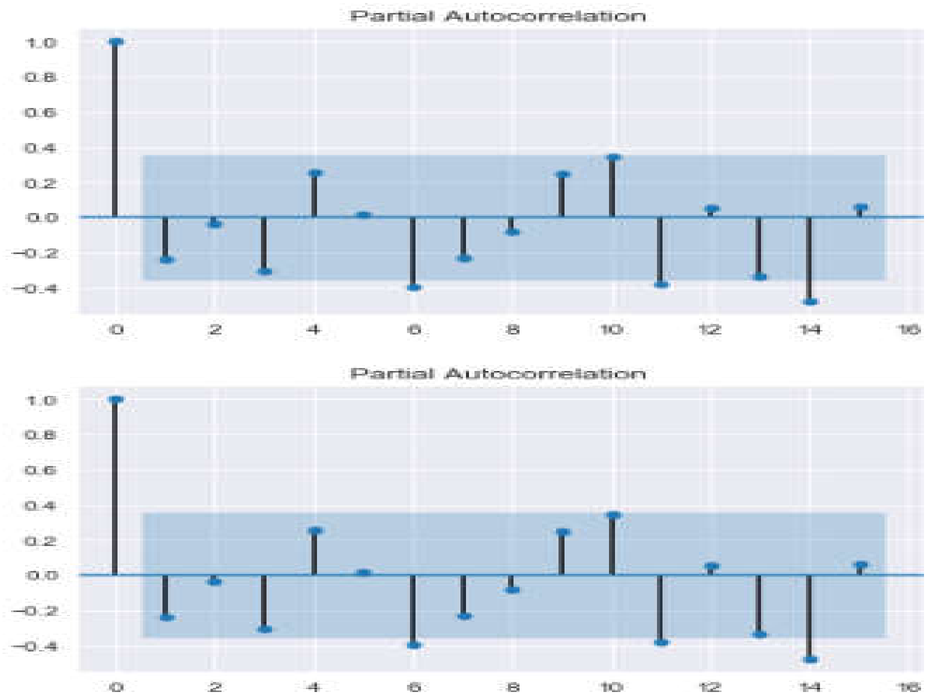

Fig 12: Partial Auto correlation function (PACF)

16. Converting data to stationary
    data_diff=data_new.diff(periods=1)
    data_diff=data_diff[1:]

17. Plot acf and pacf for the new data

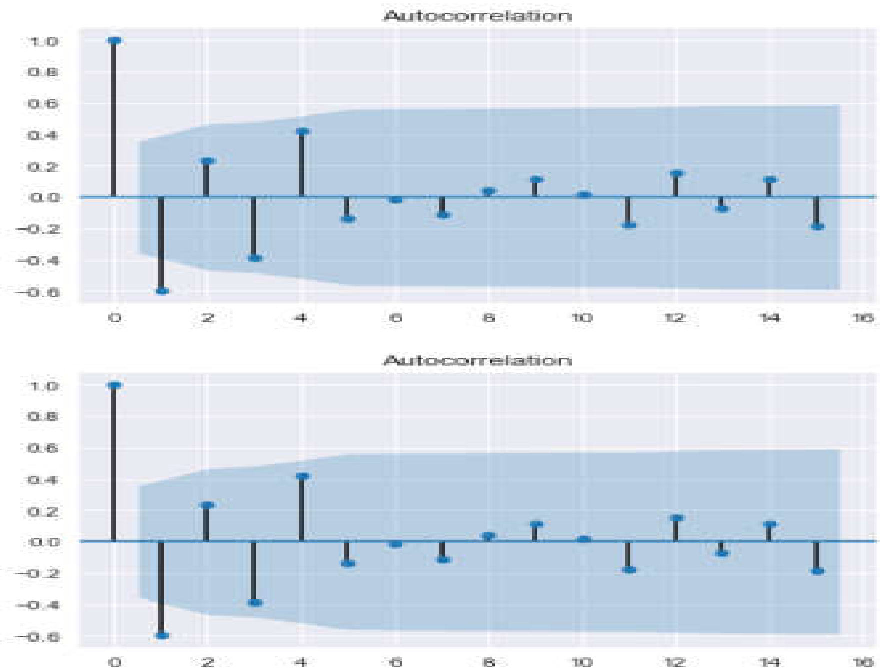

Fig 13: Auto correlation function (ACF)

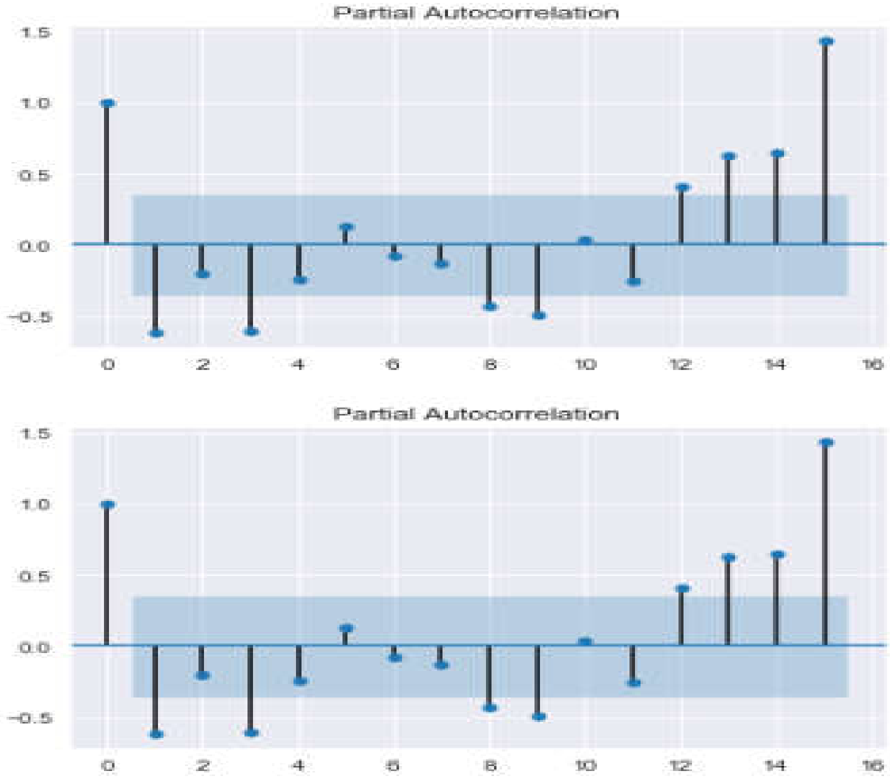

Fig 14: Partial Auto correlation function (PACF)

18. Plot the line plot for new data

Fig 15: Line plot representing orders on demand with respect to Job Card Date for data_diff dataframe

Next, different kinds of time series forecasting algorithms were being used. They are as follows:

19. **AutoRegressive(AR) Model**

1) Initialize test and train data and declare predictions
   X=data_diff.values
   train=X[0:450]
   test=X[451:]
   predictions=[]

2) Import the necessary packages for AR model and define the model
   from statsmodels.tsa.ar_model import AR
   from sklearn.metrics import mean_squared_error

model_ar=AR(train)
model_ar_fit=model_ar.fit()
print(model_ar_fit.aic)

3) Define the value of predictions
predictions=model_ar_fit.predict(start=50,end=150)
predictions

4) Plot the line graph for AR model consisting of actual data and predicted data .

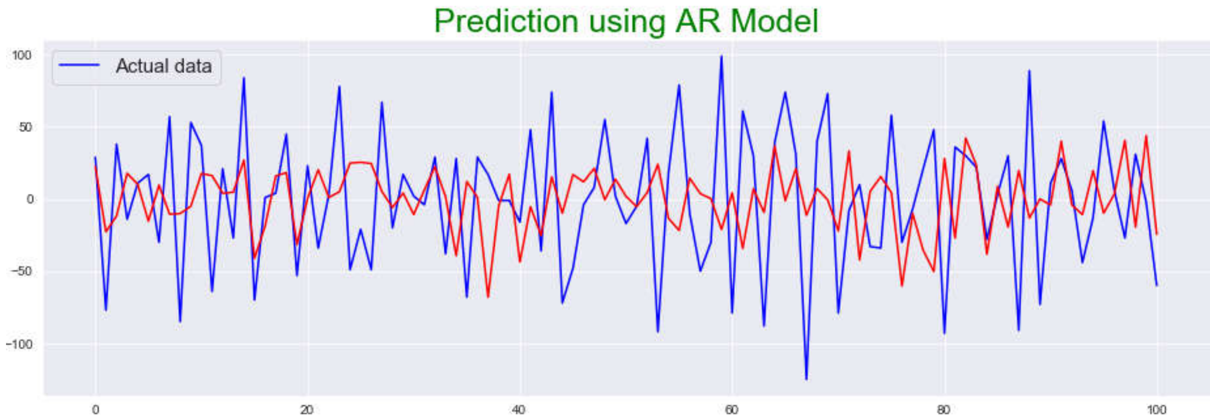

Fig 16: Prediction using AR Model

5) Find out the mean squared error
mean_squared_error(test,predictions)
np.sqrt(mean_squared_error(test,predictions))

20. <u>**Moving Average (MA) Model**</u>

1) Initialize test and train data and declare predictions
X=data_diff.values
train=X[0:450]
test=X[451:]
predictions=[]

2) Import the necessary packages for MA model and define the model
from statsmodels.tsa.arima_model import ARMA
from sklearn.metrics import mean_squared_error
model_ma=ARMA(train, order=(0, 1))
model_ma_fit=model_ma.fit()
print(model_ma_fit.aic)

3) Define the value of predictions
predictions=model_ar_fit.predict(start=50,end=150)
predictions

4) Plot the line graph for MA model consisting of actual data and predicted data.

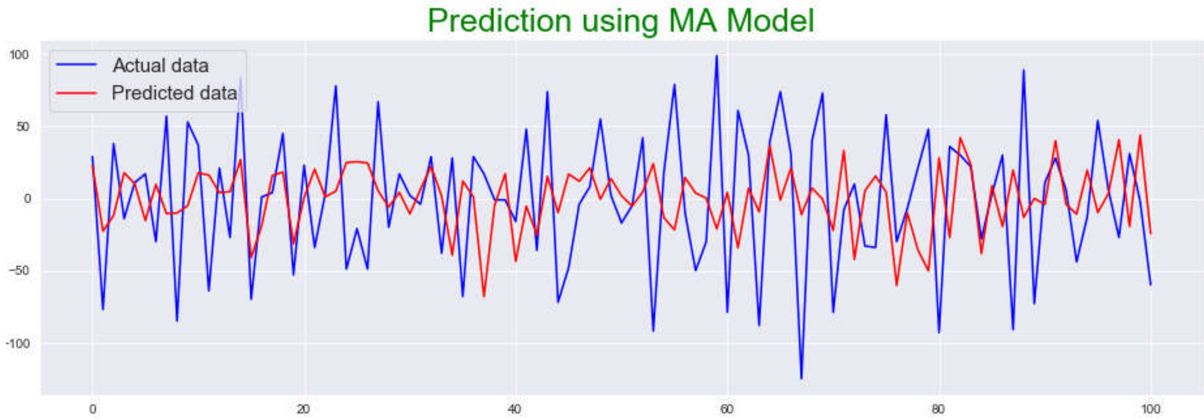

Fig 17: Prediction using MA Model

5) Find out the mean squared error
   mean_squared_error(test,predictions)
   np.sqrt(mean_squared_error(test,predictions))

21. **<u>Auto Regressive Moving Average (ARMA) Model</u>**

1) Initialize test and train data and declare predictions
   X=data_diff.values
   train=X[0:450]
   test=X[451:]
   predictions=[]

2) Import the necessary packages for ARMA model and define the model
   from statsmodels.tsa.arima_model import ARMA
   from sklearn.metrics import mean_squared_error
   model_arma=ARMA(train,order=(1,1))
   model_arma_fit=model_arma.fit()
   print(model_arma_fit.aic)

3) Define the value of predictions
   predictions=model_arma_fit.predict(start=50,end=150)
   predictions

4) Plot the line graph for ARMA model consisting of actual data and predicted data.

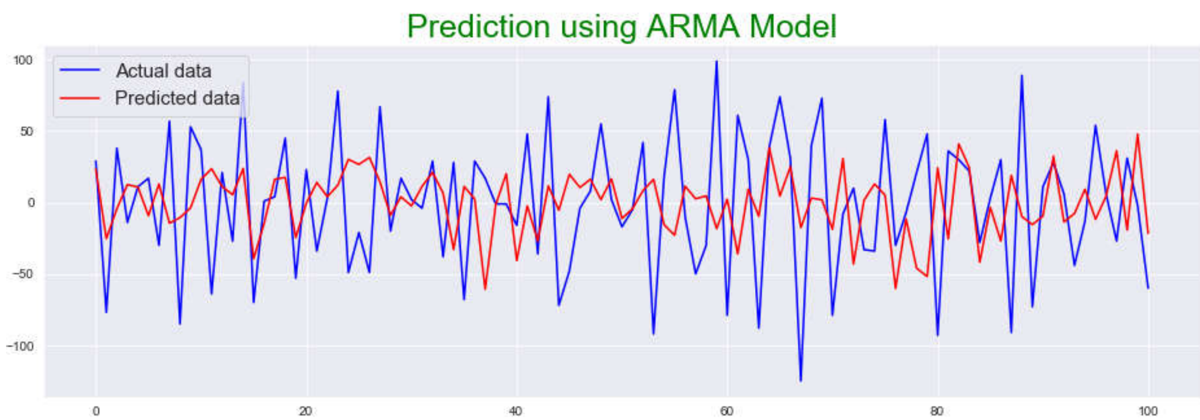

Fig 18: Prediction using ARMA Model

5) Find out the mean squared error
   mean_squared_error(test,predictions)

np.sqrt(mean_squared_error(test,predictions))

22. **Auto Regressive Integrated Moving Average (ARIMA) Model**

1) Initialize test and train data and declare predictions
X=data_diff.values
train=X[0:450]
test=X[451:]
predictions=[]

2) Import the necessary packages for ARIMA model and define the model
from statsmodels.tsa.arima_model import ARIMA
model_arima=ARIMA(test, order=(2,1,0))
model_arima_fit=model_arima.fit()
print(model_arima_fit.aic) # Akaike Information Criteria

3) Define the value of predictions
predictions=model_arima_fit.predict(start=50,end=100)
predictions

4) Plot the line graph for ARIMA model consisting of actual data and predicted data.

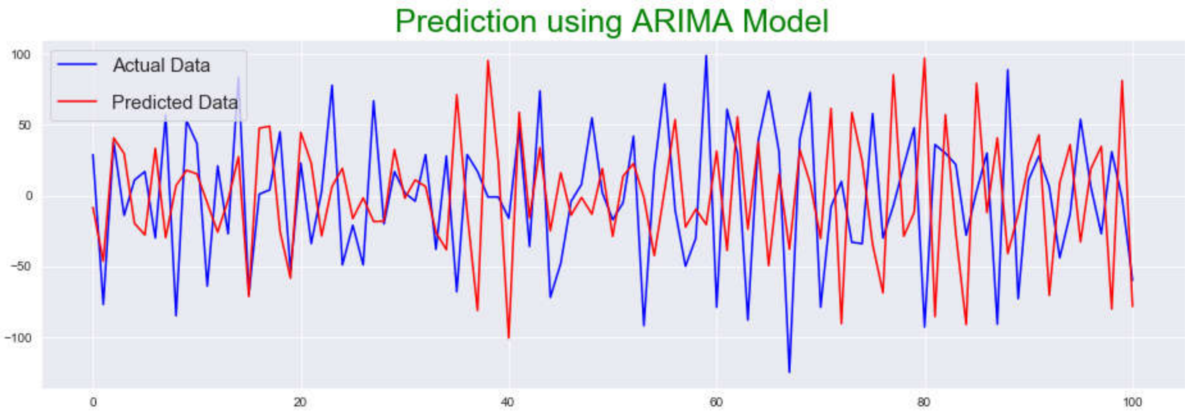

Fig 19: Prediction using ARIMA Model

5) Find out the mean squared error.
mean_squared_error(test,predictions)
np.sqrt(mean_squared_error(test,predictions))

23. **Seasonal Auto Regressive Integrated Moving Average (SARIMA) Model**

1) Initialize test and train data and declare predictions.
X=data_diff.values
train=X[0:500]
test=X[501:]
predictions=[]

2) Import the necessary packages for SARIMA model and define the model
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.metrics import mean_squared_error
model_sarima = SARIMAX(test, order=(10,2,2), seasonal_order=(1,1,1,1))
model_sarima_fit = model_sarima.fit()
print(model_sarima_fit.aic)

3) Define the value of predictions.
predictions=model_sarima_fit.predict(start=50,end=100)
predictions

4) Plot the line graph for SARIMA model consisting of actual data and predicted data.

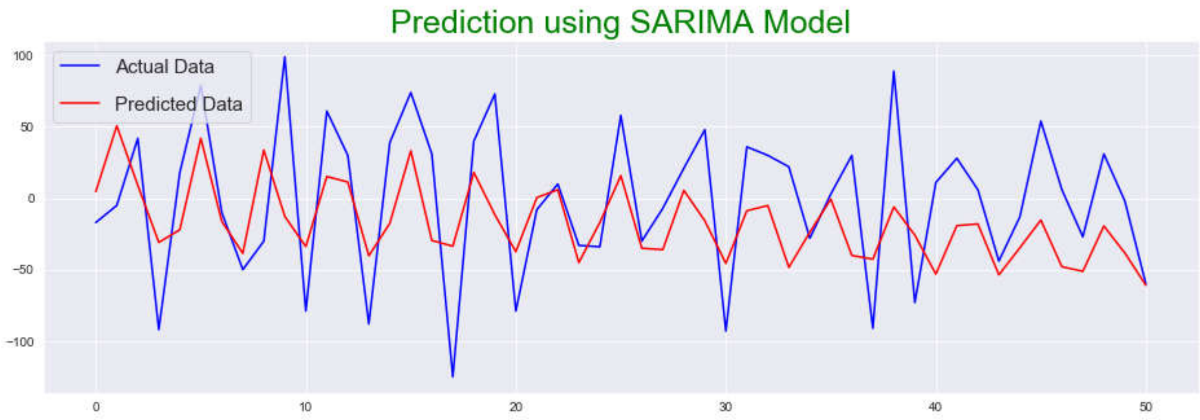

Fig 20: Prediction using SARIMA Model

5) Find out the mean squared error
mean_squared_error(test,predictions)
np.sqrt(mean_squared_error(test,predictions))

24. <u>Seasonal Auto Regressive Integrated Moving Average with Exogenous Regressors (SARIMAX) Model</u>

1) Initialize test and train data and declare predictions.
X=data_diff.values
train=X[0:400]
test=X[401:]
predictions=[]

2) Import the necessary packages for SARIMAX model and define the model
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.metrics import mean_squared_error
model_sarimax = SARIMAX(train,exdog=test,order=(3,2,2), seasonal_order=(1,1,1,1))
model_sarimax_fit = model_sarimax.fit()
print(model_sarimax_fit.aic)

3) Define the value of predictions.
predictions=model_sarimax_fit.predict(start=50,end=200)
predictions

4) Plot the line graph for SARIMAX model consisting of actual data and predicted data.

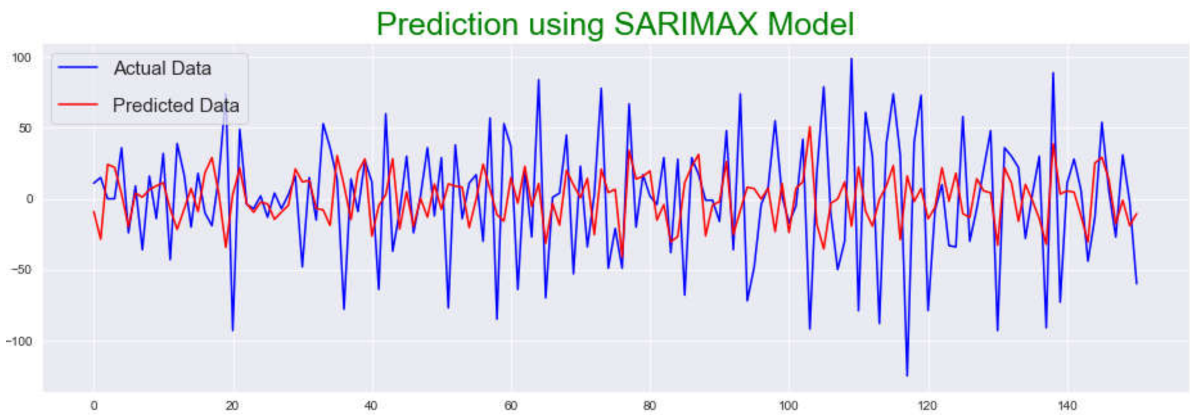

Fig 21: Prediction using SARIMAX Model

6) Find out the mean squared error
   mean_squared_error(test,predictions)
   np.sqrt(mean_squared_error(test,predictions))

Conclusion

In this project, different time series forecasting algorithms were used for finding the better predictions.
The algorithms used were Auto Regression (AR), Moving Average (MA), Auto Regressive Integrated Moving Average (ARIMA), Seasonal Auto Regressive Integrated Moving Average (SARIMA), and Seasonal Auto Regressive Integrated Moving Average with Exogenous Regressors (SARIMAX).
As per the graph, ARIMA model was giving better prediction as compared to all the other models.