Time Series Analysis and Sales Forecasting for Automotive using IBM Services

Submitted by

Team No. 1

Hitesh Agrawal

(19BAI10030)

Shiv Taneja

(19BAI10039)

Krishan Kumar Gupta

(19BAI10114)

Shrutika Nikhar

(19BCY10107)

1. INTRODUCTION

1.1 OVERVIEW

Time-series Sales forecasting has proved to be a crucial topic in every business, helping to process data taken over a long period of time. Stock exchange, logistics, retail are classic industries where the ability to build predictive models becomes a crucial differentiator in a highly-competitive business environment. With our project we have tried to apply time series analysis and sales forecasting method for Automotive using IBM cloud services: finding hidden patterns, detecting trends in sales over the years, and predicting sales in the future.

1.2 PURPOSE

Unknown Demand for vehicles is a significant problem that decreases the productivity of the production environment. Thus, forecasting plays a vital role in planning the production and inventory of an industry. Time series forecasting is a type of forecasting in which last period data is used to determine the industry's trends and sales in the future.

In this project, we are building a system that analyses the previous trends of sales which includes sales on various days and predicts future sales. The goal of this project is to forecast the sales of stores by using time series analysis. Here time series analysis algorithms such as RNN (Recurrent Neural Network) & LSTM (Long Term Short Memory) are used to analyse the past trends of sales of stores. Create and deploy flask-based web Application and integrate AI model to it.

The objective of the project is to build a web application where the user gives the last ten days' sales values and gets the prediction for the 11 th day which is showcased on UI.

2. LITERATURE SURVEY

2.1. EXISTING PROBLEM

1. Seher Arslankya in his research paper implemented time series analysis and artificial neural network methods to estimate sales for future months for the leading company in the automotive industry in Turkey. He examined the monthly data between January 2011 and July 2016 using multiple regression, moving average and artificial neural network model. Furthermore, he compared MAPE values for all models, which resulted in the ANN model giving the best result.

2. Arnis Kirshners did a comparative analysis of short time series processing methods intending to scrutinize these methods' ability to be used when analyzing short time series. The author has analyzed the moving average Method, exponential smoothening, and exponential smoothening with development trends resulting from moving average having the smallest squared error value but with large forecast smoothening for initial data.

3. Shamsul Masum elucidates on time series forecasting and its classification and approaches and strategies of time series forecasting. Furthermore, the author has demonstrated how an inappropriate point to point rolling forecast strategy leads to unrealistic outcomes and supports his argument with a comparative analysis of two

strategies using the ARIMA model. The author has concluded with the result of rolling single point outcome being deceptive for Euro Dollar Exchange Rates case considered.

4. Tamal Datta Chaudhuri proposed six different forecasting methods for predicting the time series index of the healthcare sector. The author has demonstrated a decomposition approach of time series for data from January 2010 to December 2016 and illustrated how the decomposition results provide us with useful insights into the behavior and properties exhibited by time series. The author observed that results from the ARIMA model with a horizon of 12 months came out to be the best model with the lowest RSME value, while the Holt Winters method with a horizon of 12 months has the highest RSME value.

5. Jaydip Sen, in his research paper, uses the Time series - decomposition based Method to analyze the past of the Indian realty sector and predict its future. He uses time series forecasting methods in R programming language to determine future results. He uses time series index value data of the Indian realty sector for six years from 2010-2016 month wise. The methods used for accurate predictions are: Holt Winters exponential smoothening and Autoregressive Integrated Moving Average. He analyses the results from the above-mentioned broad concepts and observes, which is the best one. With the result obtained, he argues that these can be immensely useful for portfolio managers and stock traders to buy or sell stocks at the correct time.

6. William R Huss implements univariate estimation techniques such as Holt winter exponential smoothing, Multiple regression, Linear regression to study the load on 49 largest electric utilities in the United States and forecast the load for future planning. The author uses electric utility data from 1972 to 1982, and the results indicate that for shorter periods, Holt Winters Exponential smoothing method is highly accurate, and for more extended periods, extrapolation of Linear regression horizons proves to be efficient.

7. Deepa, in this paper, reviews, and forecasts the Indian Motorcycle market using Time series forecasting. The author uses SARIMA (Seasonal autoregressive Integrated Moving Method) and Holt Winters Method for prediction. The author compares several years' data and uses MAE and MAPE method to determine which model is more accurate and concludes that both the models are pretty significant but Holt Winters method is numerically more precise than the SARIMA model. According to the author the studies can be further enhanced by using more such models.

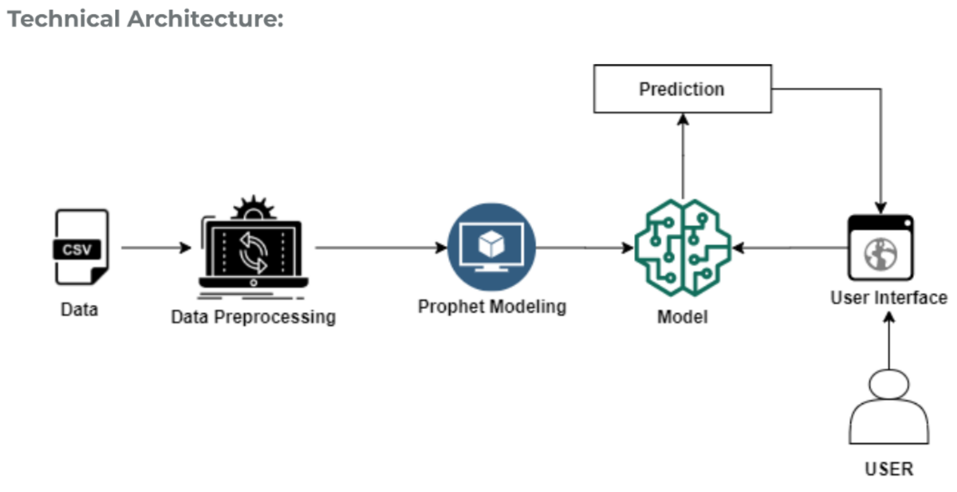
2.2. PROPOSED SOLUTION

The automotive sector is one of the developing sectors, where the most massive investments are made. Today India's automotive industry is expected to reach Rs 16 to 18 trillion by 2026. Therefore, it is crucial for companies in this sector to correctly manage their resources. To do that, Our Approach is:

1. Installation of Prerequisites.
    - Installation of Anaconda IDE / Anaconda Navigator.
    - Installation of Python packages.
2. Data Collection.
    - Create or Collect the dataset.
3. Data Pre-processing.
    - Importing of Libraries.
    - Importing of Dataset & Visualisation.
4. Model Building.
    - Fitting the prophet library.
    - Evaluation of the model.
    - Save the model.
5. Application Building

3. THEORITICAL ANALYSIS

3.1. BLOCK DIAGRAM



3.2. HARDWARE / SOFTWARE DESIGNING

Software requirements

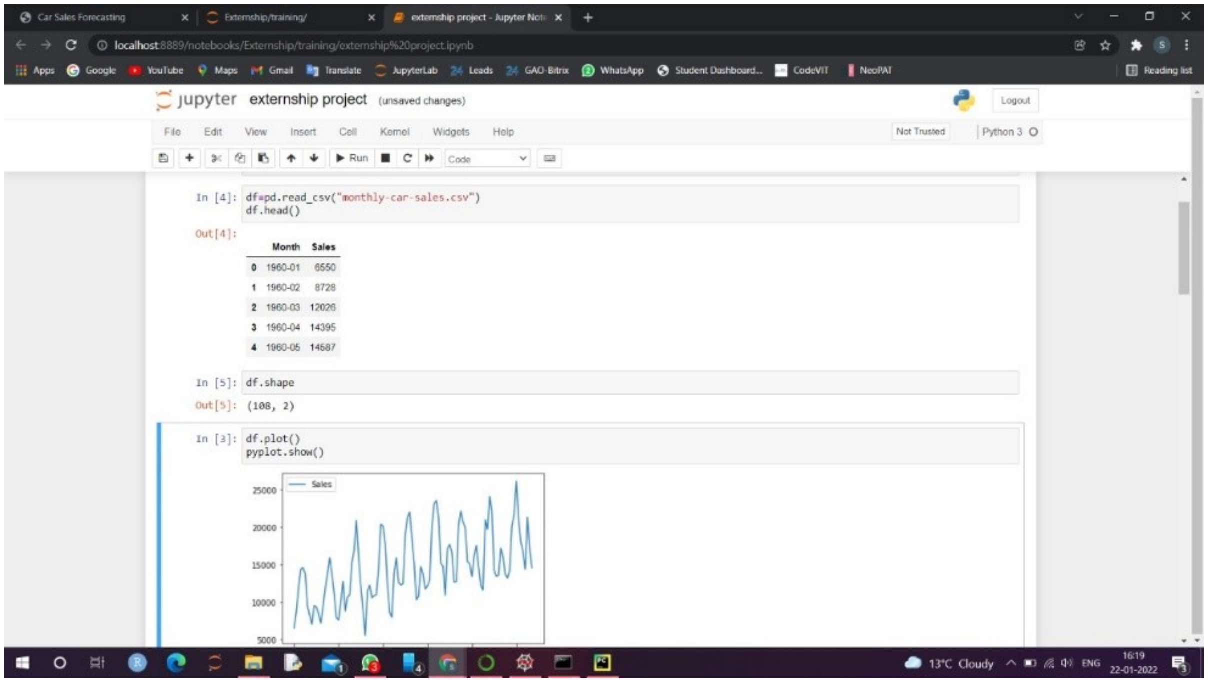
- Anaconda – Python, Jupyter notebook, Spyder
- Windows/ Linux/ Mac os
- Tensorflow
- Opencv
- Numpy
- FbProphet
- IBM cloud

Hardware requirements

- TPU
- Processor: Minimum 1 GHz
- RAM: Minimum 8 Gb
- Intel core 5 or 7 or amd 5

4. EXPERIMENTAL INVESTIGATIONS

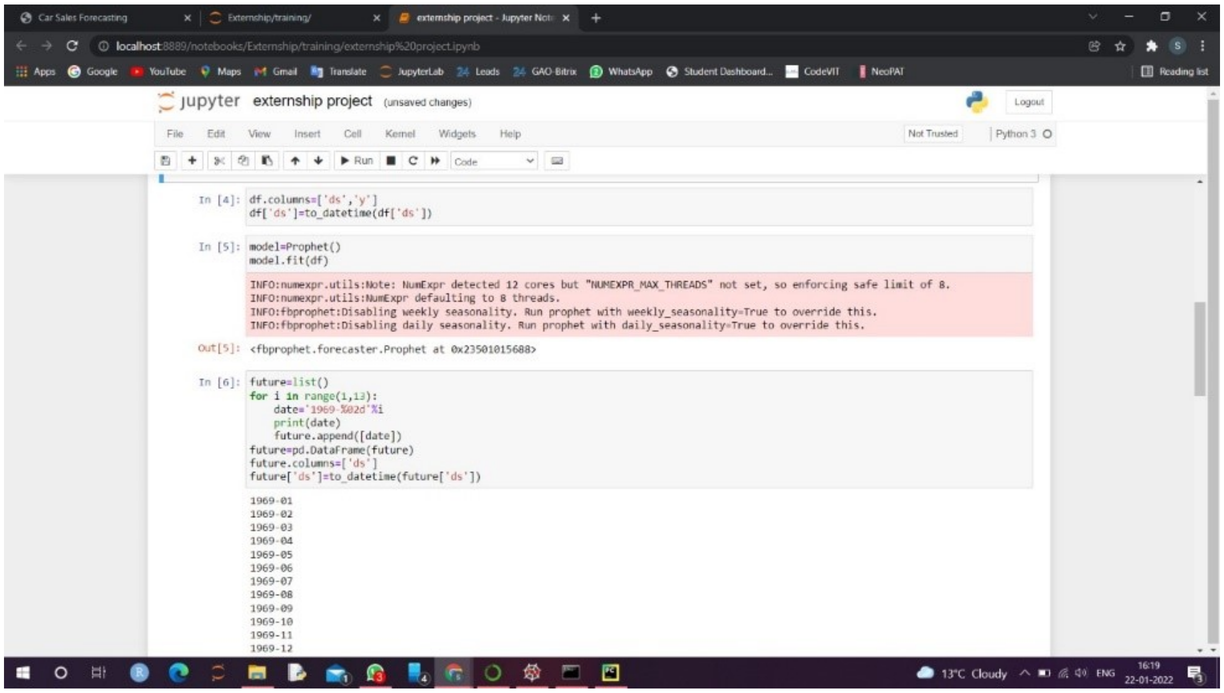
a)



jupyter
externship project (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code

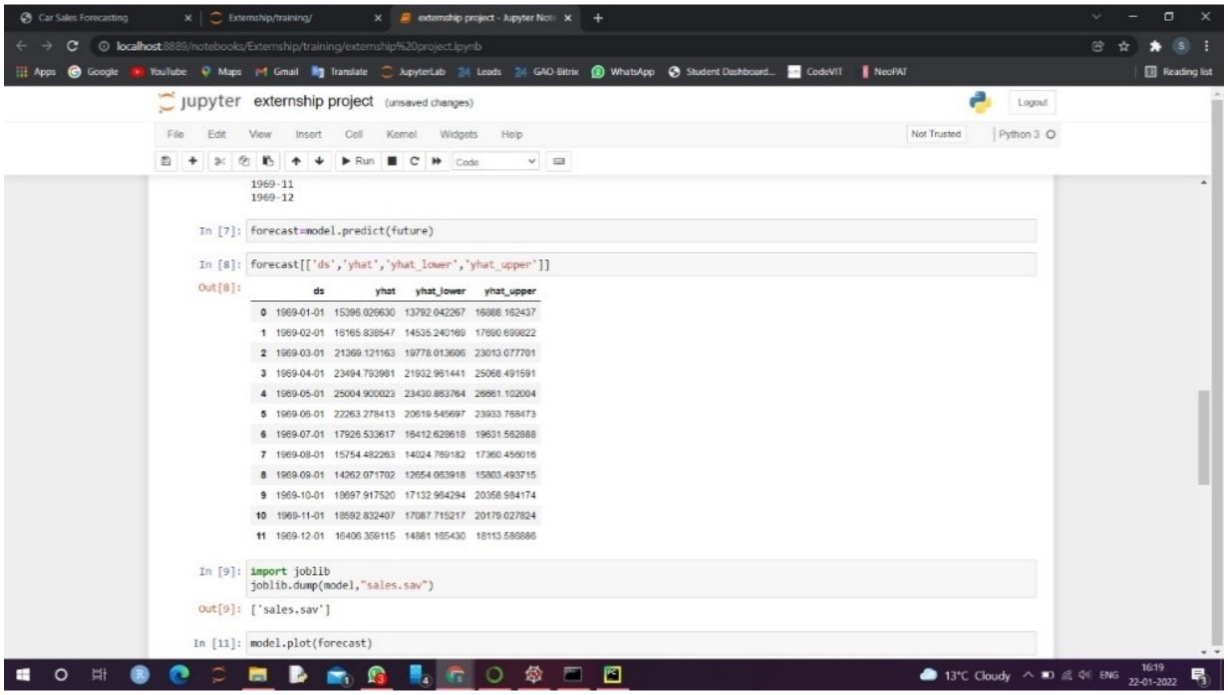
```
In [4]: df=pd.read_csv("monthly-car-sales.csv")
        df.head()
Out[4]:
           Month  Sales
        0  1960-01   6550
        1  1960-02   8728
        2  1960-03  12026
        3  1960-04  14395
        4  1960-05  14587

In [5]: df.shape
Out[5]: (108, 2)

In [3]: df.plot()
        pyplot.show()
```

b)



```
In [4]: df.columns=['ds','y']
        df['ds']=to_datetime(df['ds'])

In [5]: model=Prophet()
        model.fit(df)

        INFO:numexpr.utils:Note: NumExpr detected 12 cores but "NUMEXPR_MAX_THREADS" not set, so enforcing safe limit of 8.
        INFO:numexpr.utils:NumExpr defaulting to 8 threads.
        INFO:fbprophet:Disabling weekly seasonality. Run prophet with weekly_seasonality=True to override this.
        INFO:fbprophet:Disabling daily seasonality. Run prophet with daily_seasonality=True to override this.

Out[5]: <fbprophet.forecaster.Prophet at 0x23501015688>

In [6]: future=list()
        for i in range(1,13):
            date='1969-%02d'%i
            print(date)
            future.append([date])
        future=pd.DataFrame(future)
        future.columns=['ds']
        future['ds']=to_datetime(future['ds'])

        1969-01
        1969-02
        1969-03
        1969-04
        1969-05
        1969-06
        1969-07
        1969-08
        1969-09
        1969-10
        1969-11
        1969-12
```

c)

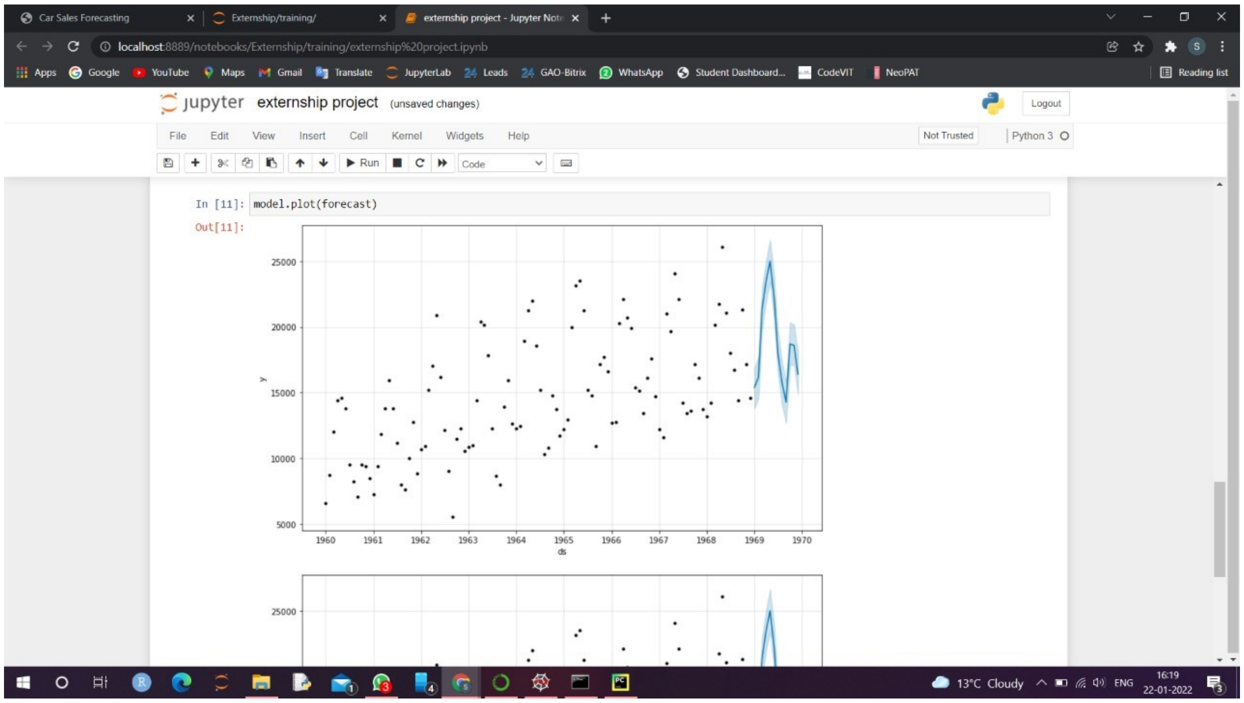


```
        1969-11
        1969-12

In [7]: forecast=model.predict(future)

In [8]: forecast[['ds','yhat','yhat_lower','yhat_upper']]

Out[8]:
```

	ds	yhat	yhat_lower	yhat_upper
0	1969-01-01	15396.026630	13792.042267	16888.162437
1	1969-02-01	16165.838547	14535.240169	17690.690822
2	1969-03-01	21369.121163	19778.013606	23013.077701
3	1969-04-01	23494.793981	21932.951441	25066.491591
4	1969-05-01	25004.900023	23430.863764	26661.102004
5	1969-06-01	22263.278413	20619.545697	23933.768473
6	1969-07-01	17926.533617	16412.628618	19631.562888
7	1969-08-01	15754.482263	14024.769182	17360.456016
8	1969-09-01	14262.071702	12654.063918	15803.493715
9	1969-10-01	18697.917520	17132.984294	20358.984174
10	1969-11-01	18592.832407	17087.715217	20179.027824
11	1969-12-01	16406.359115	14881.165430	18113.586886

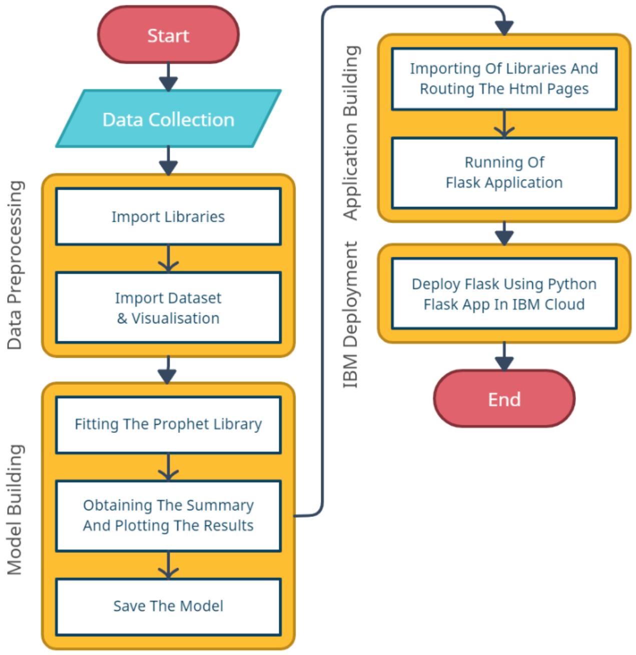
```
In [9]: import joblib
        joblib.dump(model,"sales.sav")

Out[9]: ['sales.sav']

In [11]: model.plot(forecast)
```

d)



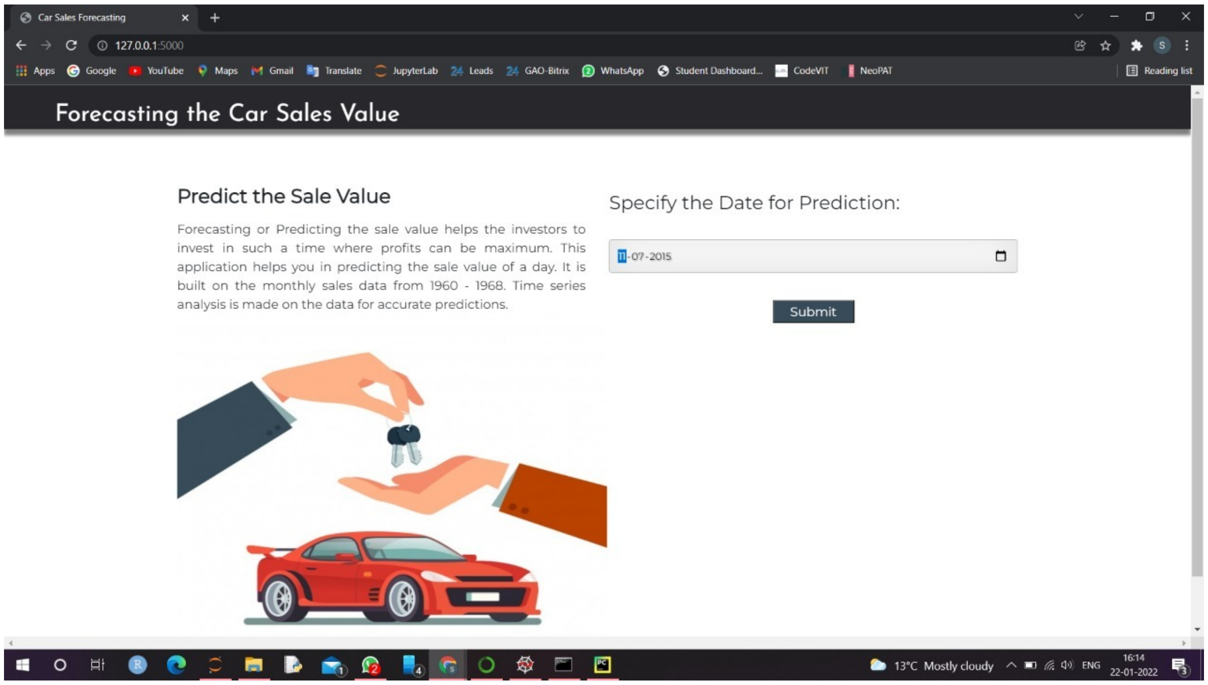
```
In [11]: model.plot(forecast)
Out[11]:
```

5. FLOWCHART


Start
Data Collection
Data Preprocessing
Import Libraries
Import Dataset & Visualisation
Model Building
Fitting The Prophet Library
Obtaining The Summary And Plotting The Results
Save The Model
Application Building
Importing Of Libraries And Routing The Html Pages
Running Of Flask Application
IBM Deployment
Deploy Flask Using Python Flask App In IBM Cloud
End

6. RESULT

Forecasting or predicting the sale value helps the investors to invest in such a time where profits can be maximum. This project guides individuals who are willing to invest or buy a car and helps them in knowing the price of a day using the prophet library. Time series analysis is made on the data for accurate predictions. The final findings we got here are when the user inputs the date for prediction and submits it, we get the sale value of that particular in Rupees.



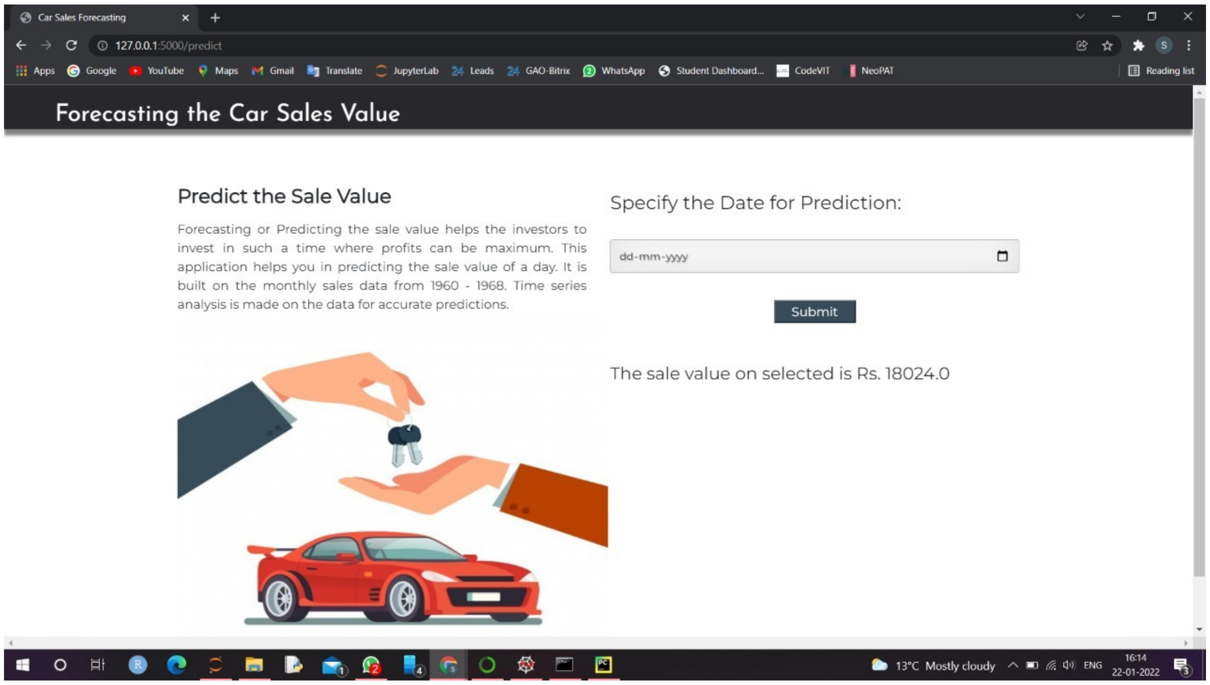
Forecasting the Car Sales Value

Predict the Sale Value

Forecasting or Predicting the sale value helps the investors to invest in such a time where profits can be maximum. This application helps you in predicting the sale value of a day. It is built on the monthly sales data from 1960 - 1968. Time series analysis is made on the data for accurate predictions.

Specify the Date for Prediction:

11-07-2015



Forecasting the Car Sales Value

Predict the Sale Value

Forecasting or Predicting the sale value helps the investors to invest in such a time where profits can be maximum. This application helps you in predicting the sale value of a day. It is built on the monthly sales data from 1960 - 1968. Time series analysis is made on the data for accurate predictions.

Specify the Date for Prediction:

dd-mm-yyyy

The sale value on selected is Rs. 18024.0

7. ADVANTAGES & DISADVANTAGES

Our solution for the Time Series Analysis and Sales Forecasting for Automotive have following advantages and disadvantages.

Advantages:

1. Time series analysis helps in identifying the patterns and also creates the opportunity to clean your data. It gives the high accuracy and provides simplicity in executing.
2. It is a really handy tool for forecasting purposes. It gives accurate predictions for future values, but it also requires more skill than regression analysis since you need to adapt your model according to the historical data.
3. It may be helpful to see how the commodity, security, or economic component shifts over time. The model is also used to analyze the possible changes in car resale value as many insurance companies depends upon this.

Disadvantages:

1. Time series analysis is useful for short-term forecasting, but it could sometimes lead to wrong predictions. This is because it requires historical data in order to construct the models, which means that if some significant changes occurred over time, then those changes will not be included within the forecasted periods.
2. Our model has been built on historical data, so it cannot be used to predict future values or trends because no one can guarantee that the historical data will remain the same as time passes.
3. The model could be over-fitted, which means that due to some random factors, a model can lead to false results.

8. APPLICATIONS

By using our solution for the Time Series Analysis and Sales Forecasting for Automotive we can do the following things:

1. A potential buyer of a used car will get a range of car resale value. So that one can buy the used car according to the range.

2. In terms of insurance, the Insured Declared Value (IDV) plays a major role. The term ‘IDV’ refers to the maximum claim, an insurer will pay if the vehicle is damaged beyond repair or is stolen.

3. Manufacturing companies will get a basic resale value range so that they can set the pricing of upcoming cars in such a way that the resale value remains high than the previous year's data of existing cars.

9. CONCLUSION

The overall purpose of the study was to prove that it’s possible to efficiently forecast car sales using a simple statistical model. During our research we were able to prove that the Decision Tree Regressor based approach has acceptable outcomes. Such models can be easily implemented with various statistical software and their computational complexity is acceptable. Also, the approach has well-studied statistical properties.

The accuracy of the predictive model for car sales forecast obtained is 87.9%. Hence it has been proved that the percentage error is not greater than 12.1 % for each of the 12 months ahead. Obviously, the accuracy of the model is high enough and the model can be used as a baseline for developing better models. The method is well suited for use in different business domains.

10. FUTURE SCOPE

Our only concern about the project is that It can only extract linear relationships within the time series data. Predictions generated may not be suitable for complex nonlinear cases. It does not efficiently extract the full relationship hidden in the data. Another limitation is that the model requires a large amount of data to generate accurate predictions. Hence these above are the two current issues that could be addressed for prospects.

11. BIBLOGRAPHY

- Project Description - https://smartinternz.com/saas-guided-project/1/forecasting-sales-of-store-using-ibm-watson-studio
- Importance of Time-series [Business Sector] - http://home.ubalt.edu/ntsbarsh/stat-data/forecast.htm
- Insight into Time-series Analysis - http://www.jetir.org/papers/JETIR2107494.pdf
- Existing Works - https://www.researchgate.net/publication/353659148_Analysis_of_Time_Series_Forecasting_Techniques_for_Indian_Automotive_Industry
- Practical Implementation and Insights - https://mindcraft.ai/concepts/time-series-analysis-and-sales-forecasting-for-automotive/

APPENDIX

- Time-series forecasting using SARIMA-based approach – https://mindcraft.ai/concepts/time-series-analysis-and-sales-forecasting-for-automotive/
- IBM academic services - https://www.youtube.com/watch?v=x6i43M7BAqE
- Source Code - https://github.com/smartinternz02/SI-GuidedProject-7195-1640667196/blob/main/Time%20Series%20Analysis%20and%20Sales%20Forecasting%20for%20Automotive%20using%20IBM%20Services/training/car_sales_ibm.ipynb

```
# coding: utf-8
# # STEP 1: DATA COLLECTION
# # STEP 2: DATA PREPROCESSING/ DATA WRANGLING
# IMPORTING LIBRARIES

import numpy as np
import pandas as pd
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0
if os.environ.get('RUNTIME_ENV_LOCATION_TYPE') == 'external':
   endpoint_9d750cc572a946a1aa22eeb94a45c58f = 'https://s3.us.cloud-object-
storage.appdomain.cloud'
else:
   endpoint_9d750cc572a946a1aa22eeb94a45c58f = 'https://s3.private.us.cloud-object-
storage.appdomain.cloud'

client_9d750cc572a946a1aa22eeb94a45c58f = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='XKj2TQISLTaxfuxxbVQnH0awhq-oqSsqxZY_maa57ysq',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url=endpoint_9d750cc572a946a1aa22eeb94a45c58f)

body = client_9d750cc572a946a1aa22eeb94a45c58f.get_object(Bucket='carseries-donotdelete-pr-
yllvezhwcxyq4x',Key='monthly-car-sales.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body )

dataset = pd.read_csv(body)
dataset.head()
dataset.tail()
dataset.head(10)
dataset['year'] = pd.DatetimeIndex(dataset['Month']).year
dataset['month'] = pd.DatetimeIndex(dataset['Month']).month
#dataset['day'] = pd.DatetimeIndex(dataset['Date']).day
dataset.head()
dataset.corr()

# Dropping Date Column, Because we already splitted the Date into Year, Month and Day
dataset.drop('Month', axis=1, inplace=True)
# Checking for null values
dataset.isnull().any()
dataset.info()
# HANDLING MISSING VALUES
import matplotlib.pyplot as plt
plt.bar(dataset['month'],dataset['Sales'],color='green')
plt.xlabel('Month')
plt.ylabel('y')
plt.title('PRICE OF CAR SALES ON THE BASIS OF MONTHS OF A YEAR')
plt.legend()
import seaborn as sns
sns.lineplot(x='year',y='Sales',data=dataset,color='red')
fig=plt.figure(figsize=(8,4))
plt.scatter(dataset['year'],dataset['Sales'],color='purple')
plt.xlabel('Month')
plt.ylabel('Price')
plt.title('PRICE OF CAR SALES ON THE BASIS OF MONTHS OF A YEAR')
plt.legend()
import seaborn as sns
sns.pairplot(dataset)
plt.show()
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
dataset['colum_name'] = le.fit_transform(dataset['column_name']

# Now, Split the dataset into X(independent variable) and Y(dependent variable)
```

```
x=dataset.iloc[:,1:3].values #inputs
y=dataset.iloc[:,0:1].values #output price only
x
y

# SPLIT THE DATA INTO TRAIN AND TEST SETS
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,
                    test_size=0.2,random_state=0)
x_train.shape
x_test.shape

# # STEP 4: BUILDING AND TESTING THE MODEL
# MULTIPLE LINEAR REGRESSION
#importing linear regression from scikit learn library
from sklearn.linear_model import LinearRegression
#mlr is object of LinearRegression
mlr=LinearRegression()
#trainig the model using fit method
mlr.fit(x_train,y_train)
y_pred=mlr.predict(x_test)
y_pred
y_test

from sklearn.metrics import r2_score
accuracy=r2_score(y_test,y_pred)
accuracy

# Note: The accuracy obtained using the Multilinear Regression Algorithm is very low...Therefore we will
not use this algorithm
# DECISION TREE REGRESSOR
#import decision tree regressor
from sklearn.tree import DecisionTreeRegressor
dtr=DecisionTreeRegressor()
#fitting the model or training the model
dtr.fit(x_train,y_train)

# PREDICTION
y_pred=dtr.predict(x_test)
y_pred
y_test

# ACCURACY EVALUATION
from sklearn.metrics import r2_score
dtraccuracy=r2_score(y_test,y_pred)
dtraccuracy

# RANDOM VALUE PREDICTION
```

```
dataset.head()
y_p=dtr.predict([[2005,12]])
y_p
y_p=dtr.predict([[1997,1]])
y_p

import pickle
pickle.dump(dtr,open('sales.pkl','wb'))
pwd

#Deployment
get_ipython().system('pip install -U ibm-watson-machine-learning')
from ibm_watson_machine_learning import APIClient
import json
import numpy as np
#Authenticate and Set space
wml_credentials = {
   "apikey":"wO8JHOKuRoeaSrIC2KyKeQORrrtlveW1DYS3-eg_31pv",
   "url":"https://us-south.ml.cloud.ibm.com"
}
wml_client = APIClient(wml_credentials)
wml_client.spaces.list()

SPACE_ID="7f302674-b2a7-49e2-93d4-fed45335a7df"
wml_client.set.default_space(SPACE_ID)
wml_client.software_specifications.list()
#Save and Deploy Model
import sklearn
sklearn.__version__
MODEL_NAME = 'car_model'
DEPLOYMENT_NAME = 'Sales_deploy'
CS_MODEL = dtr

# Set Python Version
software_spec_uid = wml_client.software_specifications.get_id_by_name('default_py3.8')

# Setup model meta
model_props = {
   wml_client.repository.ModelMetaNames.NAME: MODEL_NAME,
   wml_client.repository.ModelMetaNames.TYPE: 'scikit-learn_0.23',
   wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
}

model_details = wml_client.repository.store_model(
   model=CS_MODEL,
   meta_props=model_props,
   training_data=x_train,
   training_target=y_train
)
#model_details

model_uid = wml_client.repository.get_model_uid(model_details); model_uid
wml_client.connections.list_datasource_types()

# Set meta
deployment_props = {
    wml_client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
    wml_client.deployments.ConfigurationMetaNames.ONLINE: {}
}
# Deploy
deployment = wml_client.deployments.create(
    artifact_uid=model_uid,
    meta_props=deployment_props )
```