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:

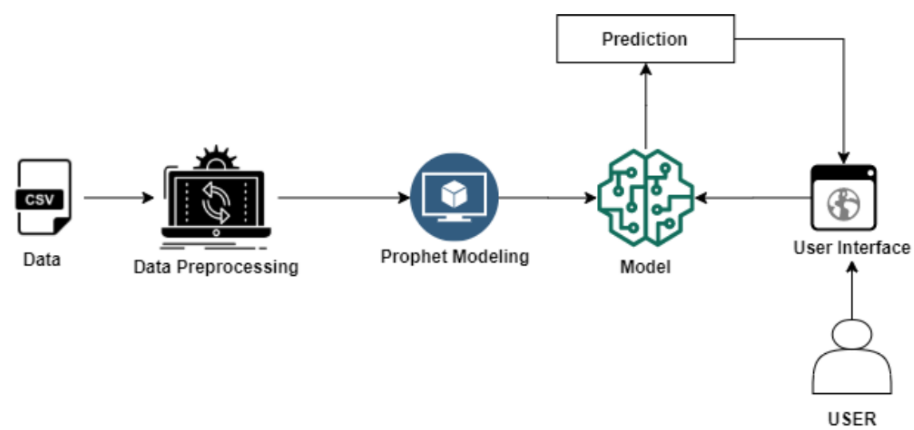
1. Installation of Prerequisites.
    - Installation of Anaconda IDE / Anaconda Navigator.
    - Installation of Python packages.
2. Data Collection.
    - Create or Collect the dataset.
3. Data Pre-processing.
    - Importing of Libraries.
    - Importing of Dataset & Visualisation.

4. Model Building.
    Fitting the prophet library.
    Evaluation of the model.
    Save the model.
5. Application Building

3. THEORITICAL ANALYSIS

3.1. BLOCK DIAGRAM

Technical Architecture:



3.2. HARDWARE / SOFTWARE DESIGNING

Software requirements

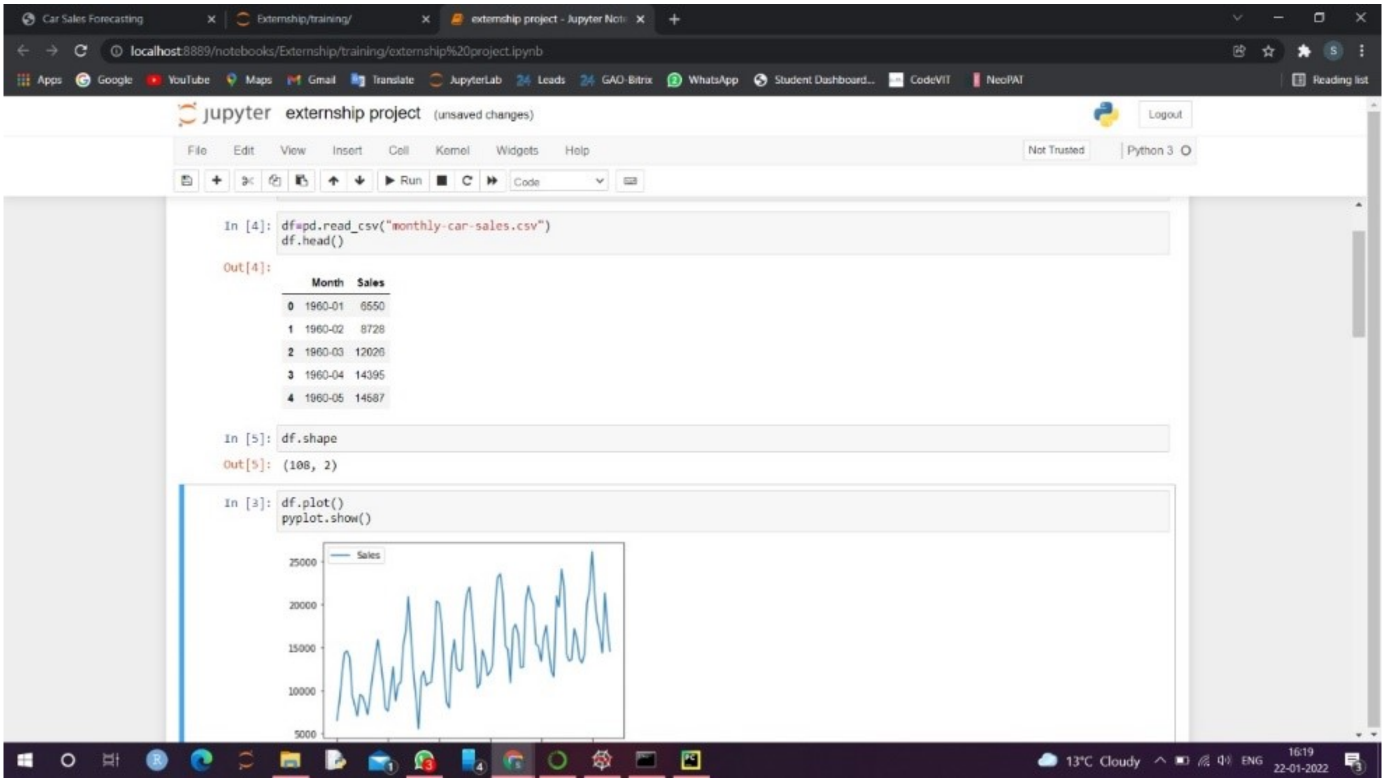
- Anaconda – Python, Jupyter notebook, Spyder
- Windows/ Linux/ Mac os
- Tensorflow
- Opencv
- Numpy
- FbProphet
- IBM cloud

Hardware requirements

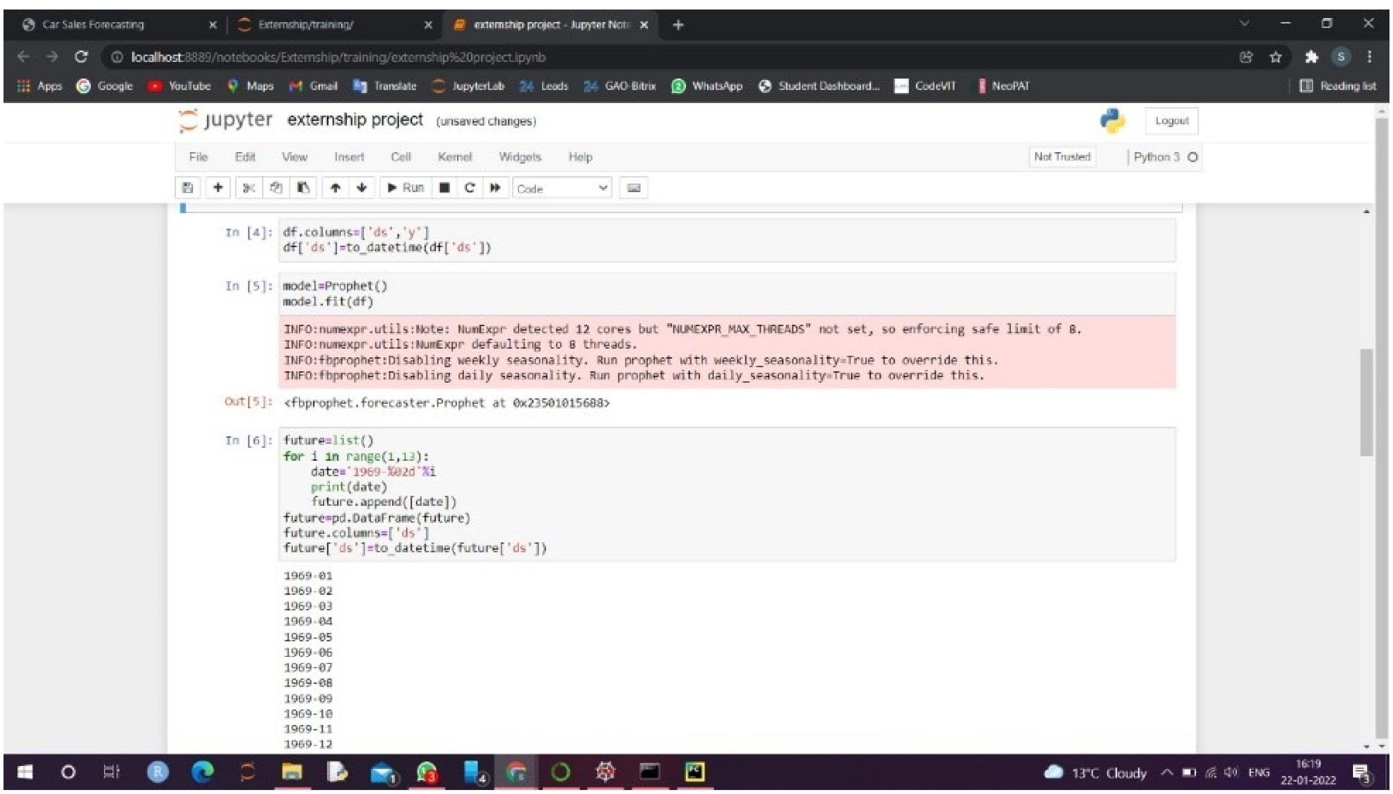
- TPU
- Processor: Minimum 1 GHz
- RAM: Minimum 8 Gb
- Intel core 5 or 7 or amd 5

4. EXPERIMENTAL INVESTIGATIONS

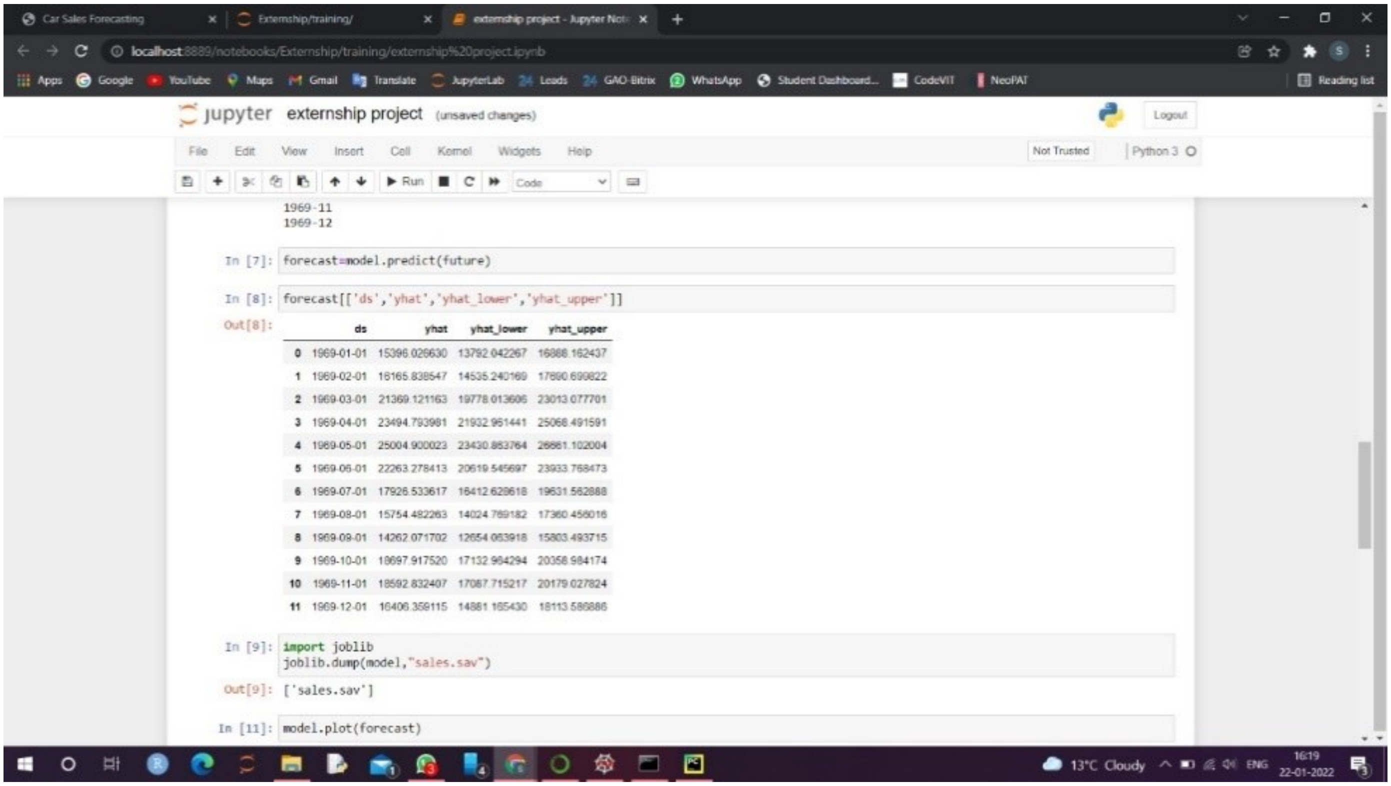
a)



b)

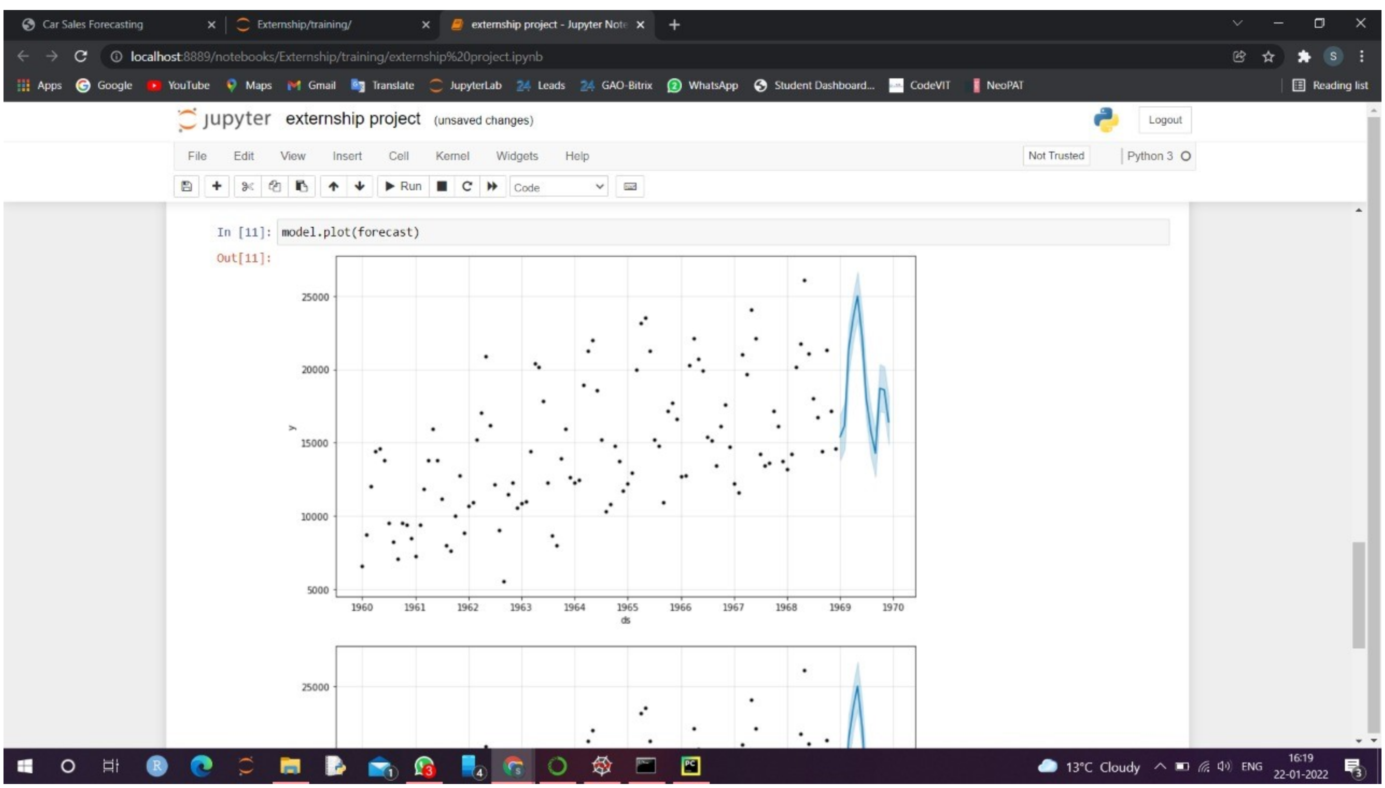


c)



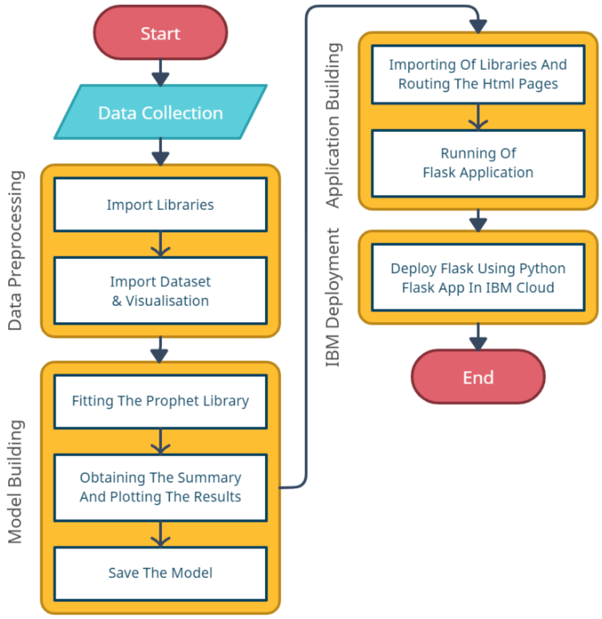
Car Sales Forecasting
Externship/training/
externship project - Jupyter Note
localhost:8889/notebooks/Externship/training/externship%20project.ipynb
jupyter externship project (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
1969-11
1969-12
In [7]: 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 17690.699822
2 1969-03-01 21369.121163 19778.013606 23013.077701
3 1969-04-01 23494.793981 21932.961441 25068.491591
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)
13°C Cloudy
ENG
16:19
22-01-2022

d)



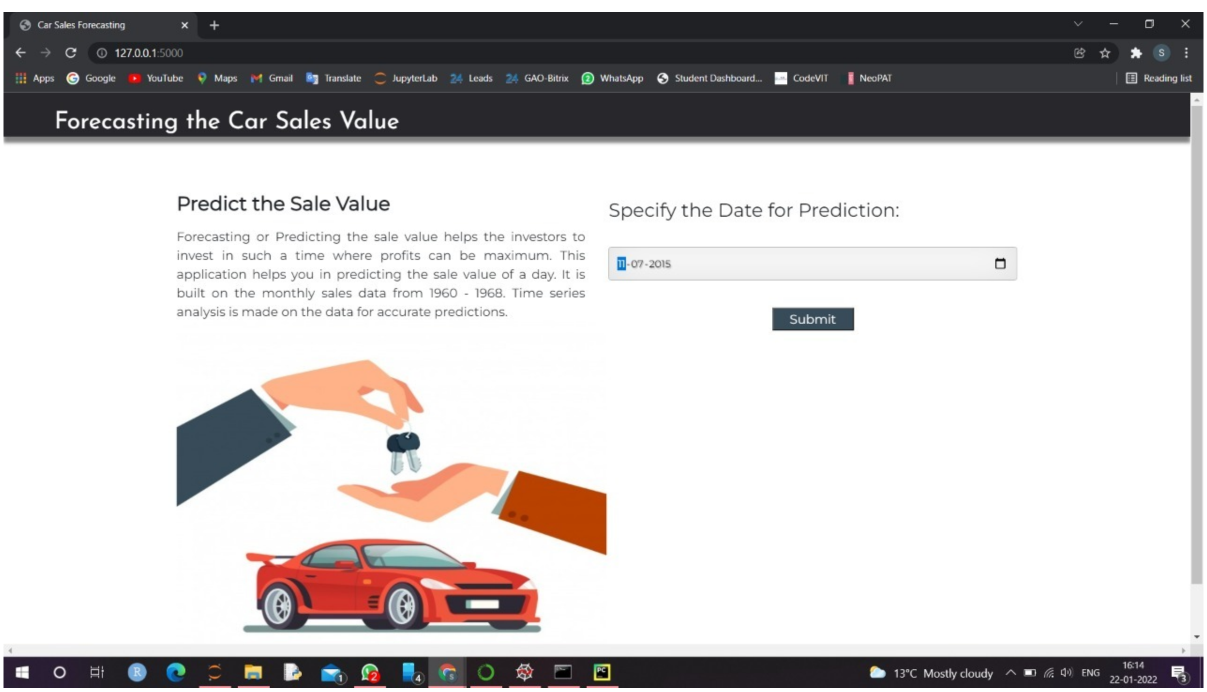
Car Sales Forecasting
Externship/training/
externship project - Jupyter Note
localhost:8889/notebooks/Externship/training/externship%20project.ipynb
jupyter externship project (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Code
In [11]: model.plot(forecast)
Out[11]:
25000
20000
15000
10000
5000
y
1960 1961 1962 1963 1964 1965 1966 1967 1968 1969 1970
ds
25000
13°C Cloudy
ENG
16:19
22-01-2022

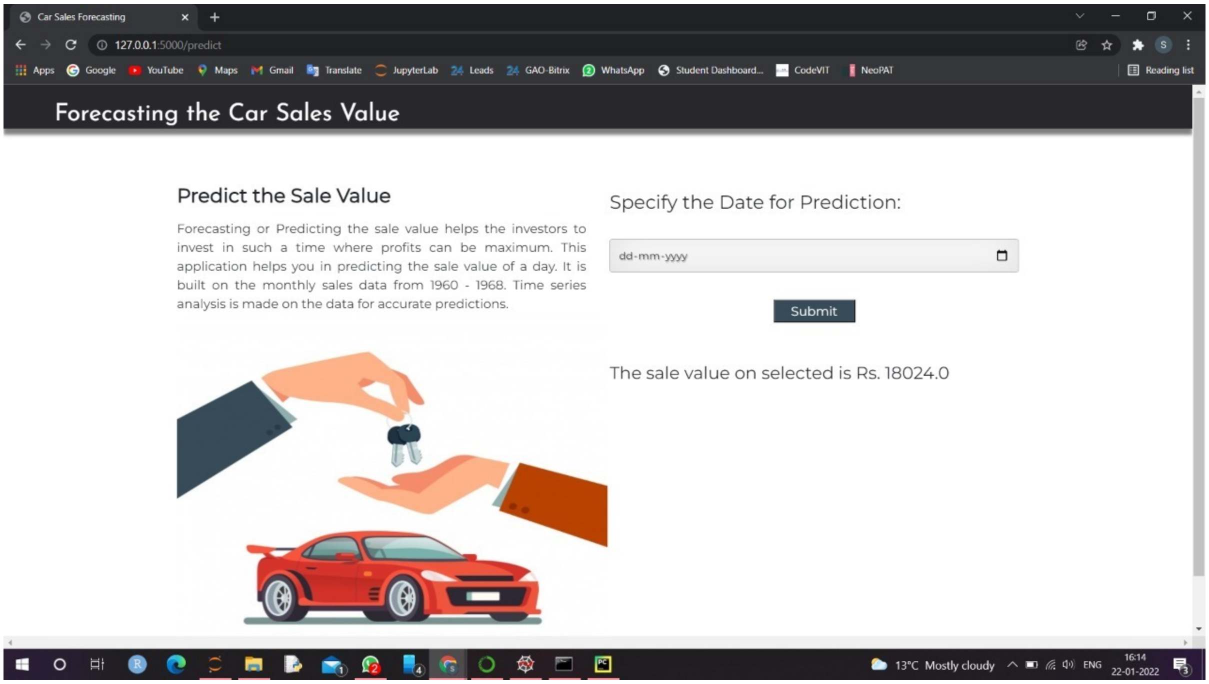
5. FLOWCHART



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.





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

Source Code Sniptes:

STEP 1: DATA COLLECTION

STEP 2: DATA PREPROCESSING/ DATA WRANGLING

IMPORTING LIBRARIES

In [1]:
```
import numpy as np
import pandas as pd
```

In [2]:
```

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

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your credentials.
# You might want to remove those credentials before you share the notebook.

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',
```

In [4]:
```
dataset.head(10)
```
Out[4]:

	Month	Sales
0	1960-01	6550
1	1960-02	8728
2	1960-03	12026
3	1960-04	14395
4	1960-05	14587
5	1960-06	13791
6	1960-07	9498
7	1960-08	8251
8	1960-09	7049
9	1960-10	9545

In [5]:
```
dataset['year'] = pd.DatetimeIndex(dataset['Month']).year
dataset['month'] = pd.DatetimeIndex(dataset['Month']).month
#dataset['day'] = pd.DatetimeIndex(dataset['Date']).day
```
In [6]:
```
dataset.head()
```
Out[6]:

	Month	Sales	year	month
0	1960-01	6550	1960	1
1	1960-02	8728	1960	2
2	1960-03	12026	1960	3
3	1960-04	14395	1960	4

In [8]:
```
dataset.drop('Month', axis=1, inplace=True)
```
Checking for null values

In [9]:
```
dataset.isnull().any()
```
Out[9]:
```
Sales    False
year     False
month    False
dtype: bool
```
In [10]:
```
dataset.info()
```
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 108 entries, 0 to 107
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Sales   108 non-null    int64
 1   year    108 non-null    int64
 2   month   108 non-null    int64
dtypes: int64(3)
memory usage: 2.7 KB
```
HANDLING MISSING VALUES

In [11]:
```
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()
```
SPLIT THE DATA INTO TRAIN AND TEST SETS

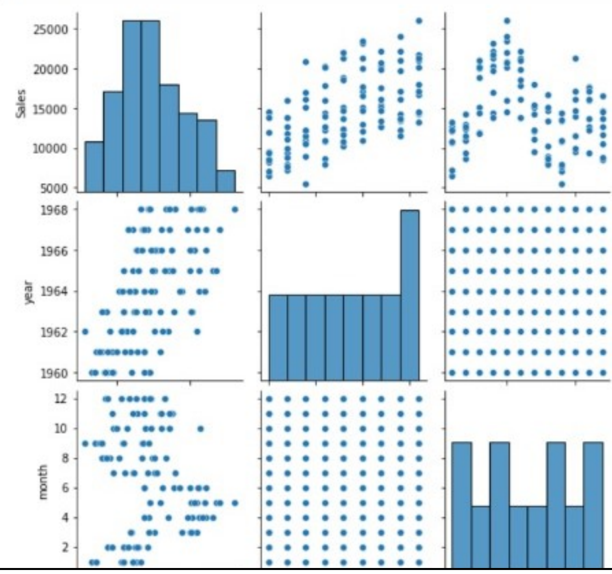
In [18]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,
                                test_size=0.2,random_state=0)
```
In [19]:
```
x_train.shape
```
Out[19]: (86, 2)

In [20]:
```
x_test.shape
```
Out[20]: (22, 2)

STEP 4: BUILDING AND TESTING THE MODEL

MULTIPLE LINEAR REGRESSION

In [21]:
```
#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)
```
Out[21]: LinearRegression()

In [22]:
```
y_pred=mlr.predict(x_test)
```
In [23]:
```
y_pred
```
In [14]:
```
import seaborn as sns
sns.pairplot(dataset)
plt.show()
```


In [25]:
```python
from sklearn.metrics import r2_score
accuracy=r2_score(y_test,y_pred)
```

In [26]:
```python
accuracy
```

Out[26]: 0.31563714627354245

Note: The accuracy obtained using the Multilinear Regression Algorithm is very low...Therefore we will not use this algorithm

DECISION TREE REGRESSOR

In [27]:
```python
#import decision tree regressor
from sklearn.tree import DecisionTreeRegressor
dtr=DecisionTreeRegressor()
#fitting the model or training the model
dtr.fit(x_train,y_train)
```

Out[27]: DecisionTreeRegressor()

PREDICTION

In [28]:
```python
y_pred=dtr.predict(x_test)
```

In [29]:
```python
y_pred
```

Out[29]:
```
array([12674.,  9545., 23125., 11837., 10862., 24081., 17180.,  7975.,
       13784., 20249., 13401., 10015., 15936., 12674., 23541., 21259.,
        8728., 12965., 12965., 12268., 13434.,  7610.])
```

In [30]:
```python
y_test
```

Out[30]:
```
array([[12225],
       [ 9364],
       [22135],
       [12026],
       [10677],
       [26099],
       [14577],
       [ 8251],
       [15926],
       [20985],
       [10895],
       [12759],
       [13932],
       [12181],
       [20677],
       [22015],
       [ 9374],
       [12760],
       [11608],
       [15175],
       [16722],
       [ 7049]])
```

ACCURACY EVALUATION

In [31]:
```python
from sklearn.metrics import r2_score
dtraccuracy=r2_score(y_test,y_pred)
```

In [32]:
```python
dtraccuracy
```

Out[32]: 0.8783651352196062

RANDOM VALUE PREDICTION

In [33]:
```python
dataset.head()
```

Out[33]:

	Sales	year	month
0	6550	1960	1
1	8728	1960	2
2	12026	1960	3
3	14395	1960	4
4	14587	1960	5

In [34]:
```python
y_p=dtr.predict([[2005,12]])
```

In [35]:
```python
y_p
```

Out[35]: array([17180.])

In [36]:
```python
y_p=dtr.predict([[1997,1]])
```

In [37]:
```python
y_p
```

Out[37]: array([13210.])

In [38]:
```python
import pickle
pickle.dump(dtr,open('sales.pkl','wb'))
```

In [39]:
```python
pwd
```

Out[39]: '/home/wsuser/work'

Deployment

URLS Dallas: https://us-south.ml.cloud.ibm.com, London - https://eu-gb.ml.cloud.ibm.com, Frankfurt - https://eu-de.ml.cloud.ibm.com, Tokyo - https://jp-tok.ml.cloud.ibm.com

Import and Install dependencies

In [40]:
```
!pip install -U ibm-watson-machine-learning
```

```
Requirement already satisfied: ibm-watson-machine-learning in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (1.0.176)
Collecting ibm-watson-machine-learning
  Downloading ibm_watson_machine_learning-1.0.178-py3-none-any.whl (1.8 MB)
     |████████████████████████████████| 1.8 MB 27.8 MB/s eta 0:00:01
Requirement already satisfied: packaging in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm-watson-machine-learning) (20.9)
Requirement already satisfied: lomond in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm-watson-machine-learning) (0.3.3)
Requirement already satisfied: certifi in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm-watson-machine-learning) (2021.10.8)
Requirement already satisfied: ibm-cos-sdk==2.7.* in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm-watson-machine-learning) (2.7.0)
Requirement already satisfied: urllib3 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm-watson-machine-learning) (1.26.6)
Requirement already satisfied: tabulate in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm-watson-machine-learning) (0.8.9)
Requirement already satisfied: requests in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm-watson-machine-learning) (2.25.1)
Requirement already satisfied: pandas<1.3.0,>=0.24.2 in /opt/conda/envs/Python-3.8-main/lib/python3.8/site-packages (from ibm-watson-machine-learning) (1.2.4)
```

Authenticate and Set space

In [42]:
```
wml_credentials = {
    "apikey":"Dnr8pZzKw-pKWsc_zxaS_5HCs9oslwvd5cFfdGrMriYE",
    "url":"https://us-south.ml.cloud.ibm.com"
}
```

In [43]:
```
wml_client = APIClient(wml_credentials)
wml_client.spaces.list()
```

```
Note: 'limit' is not provided. Only first 50 records will be displayed if the number of records exceed 50
------------------------------------  ------------  ------------------------
ID                                    NAME          CREATED
7f302674-b2a7-49e2-93d4-fed45335a7df  Sales_deploy  2022-01-21T06:53:12.796Z
------------------------------------  ------------  ------------------------
```

In [44]:
```
SPACE_ID="7f302674-b2a7-49e2-93d4-fed45335a7df"
```

In [45]:
```
wml_client.set.default_space(SPACE_ID)
```

Out[45]:
```
'SUCCESS'
```

In [46]:
```
wml_client.software_specifications.list()
```

```
-----------------------------  ------------------------------------  ----
NAME                           ASSET_ID                              TYPE
default_py3.6                  0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
pytorch-onnx_1.3-py3.7-edt     069ea134-3346-5748-b513-49120e15d288  base
scikit-learn_0.20-py3.6        09c5a1d0-9c1e-4473-a344-eb7b665ff687  base
spark-mllib_3.0-scala_2.12     09f4cff0-90a7-5899-b9ed-1ef348aebdee  base
ai-function_0.1-py3.6          0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda  base
```

Note: Only first 50 records were displayed. To display more use 'limit' parameter.

Save and Deploy Model

In [47]:
```python
import sklearn
sklearn.__version__
```

Out[47]: '0.23.2'

In [48]:
```python
MODEL_NAME = 'car_model'
DEPLOYMENT_NAME = 'Sales_deploy'
CS_MODEL = dtr
```

In [49]:
```python
# 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
}
```

In [50]:
```python
#Save model
model_details = wml_client.repository.store_model(
    model=CS_MODEL,
    meta_props=model_props,
    training_data=x_train,
    training_target=y_train
)
```


```
 'metadata': {'created_at': '2022-01-21T12:04:11.191Z',
  'id': 'df2681ac-e941-4753-89d0-068e724821ef',
  'modified_at': '2022-01-21T12:04:13.736Z',
  'name': 'car_model',
  'owner': 'IBMid-664002W3U0',
  'resource_key': 'b97743c2-d716-4b1e-baa9-7fa4c735e610',
  'space_id': '7f302674-b2a7-49e2-93d4-fed45335a7df'},
 'system': {'warnings': []}}
```


In [52]:
```python
model_uid = wml_client.repository.get_model_uid(model_details); model_uid
```

Out[52]: 'df2681ac-e941-4753-89d0-068e724821ef'

In [53]:
```python
wml_client.connections.list_datasource_types()
```

NAME	DATASOURCE_ID	TYPE	STATUS
informix	029e5d1c-ba73-4b09-b742-14c3a39b6cf9	database	active
postgresql-ibmcloud	048ed1bf-516c-46f0-ae90-fa3349d8bc1c	database	active
googlecloudstorage	05b7f0ea-6ae4-45e2-a455-cc280f110825	file	active
impala	05c58384-862e-4597-b19a-c71ea7e760bc	database	active
salesforce	06847b16-07b4-4415-a924-c63d11a17aa1	database	active
datastax-ibmcloud	0bd5946b-6fcb-4253-bf76-48b362d24a89	database	active
cosmos	0c431748-2572-11ea-978f-2e728ce88125	file	active
odbc-datastage	0ca92c3d-0e46-3b42-a573-77958d53c9be	database	active
mysql-compose	0cd4b64c-b485-47ed-a8c4-329c25412de3	database	active
hive	0fd83fe5-8995-4e2e-a1be-679bb8813a6d	database	active
cognos-analytics	11f3029d-a1cf-4c4d-b8e7-64422fa54a94	file	active
cassandra-datastage	123e4263-dd25-44e5-8282-cf1b2eeea9bd	generic	active
bluemixcloudobjectstorage	193a97c1-4475-4a19-b90c-295c4fdc6517	file	active
elasticsearch	200d71ab-24a5-4b3d-85a4-a365bdd0d4cb	file	active
webspheremq-datastage	21364ca9-5b2d-323e-bd4d-59ba961f75fb	database	active
odata	27c3e1b0-b7d2-4e32-9511-1b8aaa197de0	generic	active
azurefilestorage	2a7b4fa1-c770-4807-8871-a3c5def5aa2d	file	active
bigsql	2bdd9544-f13a-47b6-b6c3-f5964a08066a	database	active

```
mysql                     b2cc3dc2-aff7-4a80-8f80-5e8c5703e9d2  database  active
hdfs-apache               c10e5224-f17d-4524-844f-e97b1305e489  file      active
netezza                   c2a82a72-0711-4376-a468-4e9951cabf22  database  active
db2eventstore             c42bcde4-4345-4fb4-b7da-c8c557527c8b  database  active
mongodb                   c6fb9293-51eb-4f2b-b20c-4dafa3136744  database  active
db2zos                    c8d3eab2-25f6-4a90-8e10-0b4226693c45  database  active
tm1odata                  c8f3d379-78b2-4bad-969d-2e928277377e  generic   active
cassandra                 e6ff8c10-4199-4b58-9a93-749411eafacd  database  active
dashdb                    cfdcb449-1204-44ba-baa6-9a8a878e6aa7  database  active
custom-noop               dca613ef-5e34-4eca-9a80-fedcf9122834  generic   system
ftp                       d5dbc62f-7c4c-4d49-8eb2-dab6cef2969c  file      active
db2-datastage             fa31fba9-10e9-32d7-968c-f677fffd1e3b  database  active
oracle-datastage          dd22f798-8c9b-41fa-841e-d66cbdf50722  generic   active
postgresql                e1c23729-99d8-4407-b3df-336e33ffdc82  database  active
greenplum                 e278eff1-a7c4-4d60-9a02-bde1bb1d26ef  database  active
kafka-datastage           f13bc9b7-4a46-48f4-99c3-01d943334ba7  generic   active
mariadb                   f3ee04c2-7c3b-4534-b300-eb6ef701646d  database  active
------------------------  ------------------------------------  --------  ------
```

In [54]:
```python
# Set meta
deployment_props = {
    wml_client.deployments.ConfigurationMetaNames.NAME:DEPLOYMENT_NAME,
    wml_client.deployments.ConfigurationMetaNames.ONLINE: {}
}
```

In [ ]:
```python
# Deploy
deployment = wml_client.deployments.create(
    artifact_uid=model_uid,
    meta_props=deployment_props
)
```