Future Sales Prediction using Machine Learning

Phase – 2 Project Submission Document

Project :Future Sales prediction Using machine learning

Team Members

- Jensing Samuel A S
- Dilli Ganesh T
- Mageshwar S V
- Kalaivendhan A
- Devanesan R

Introduction:

Sales play a key role in the business. At the company level, sales forecasting is the major part of the business plan and significant inputs for decision-making activities. It is essential for organizations to produce the required quantity at the 7 specified time.

For that, sales forecasting will gives the idea about how an organization should manage its budgeting, workforce and resources.

This forecasting helps the business management to determine how much products should be manufacture, how much revenue can be expected and what could be the requirement of employees, investment and equipment. By analyzing the future trends and needs, Sales forecasting helps to improve the business growth.

The traditional forecasting systems have some drawbacks related to accuracy of the forecasting and handling enormous amount of data. To overcome this problem, Machine-Learning (ML) techniques have been discovered. These techniques helps to analyses the bigdata and plays a important role in sales forecasting. Here we have used supervised machine learning techniques for the sales forecasting

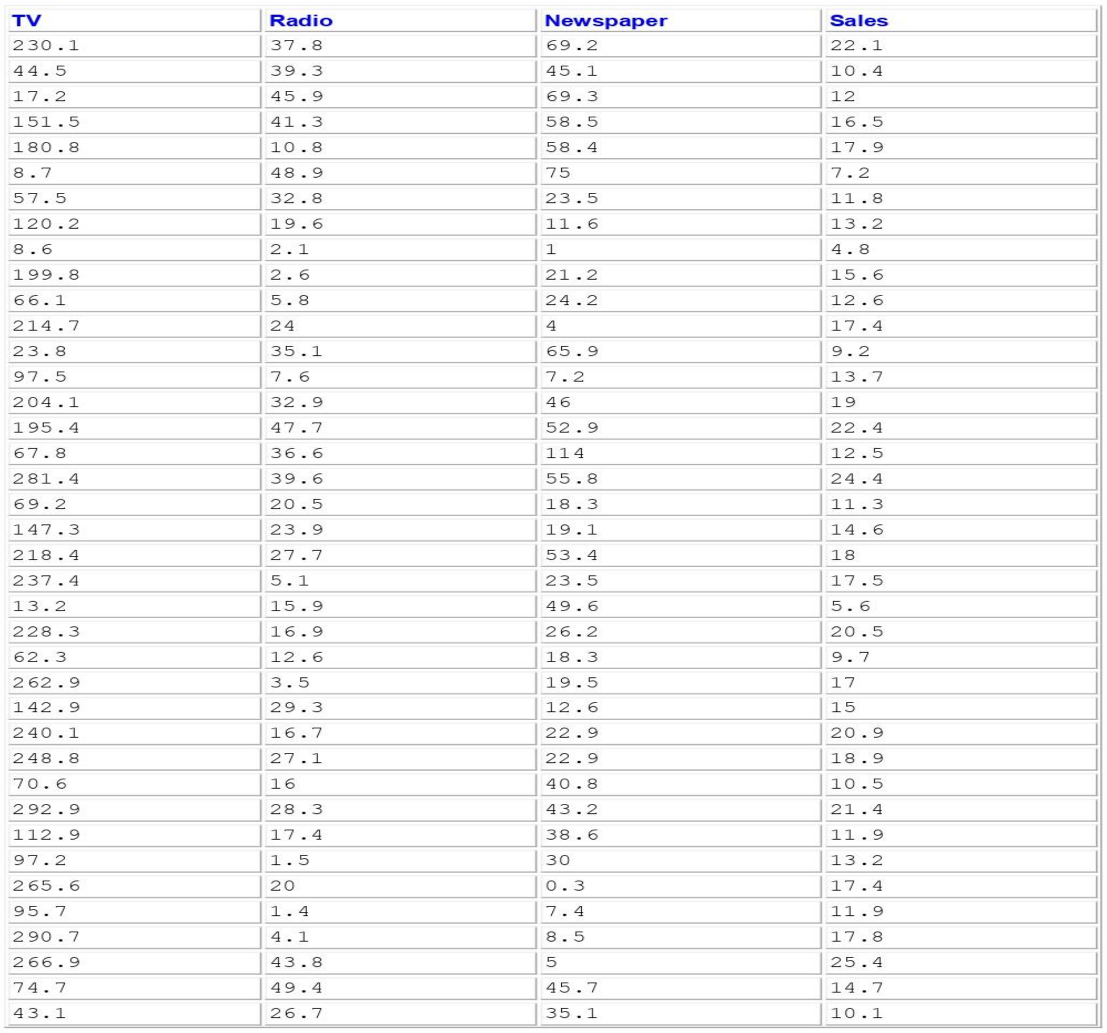
Content for Project Phase 2 :

Consider exploring more advanced time series forecasting techniques like Prophet or LSTM networks for improved accuracy in predicting future sales.

Data Source

A good data source for house price prediction using machine learning should beAccurate, Complete, Covering the geographic area of interest, Accessible.

Dataset Link: https://www.kaggle.com/datasets/chakradharmattapalli/future-sales-prediction



TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	12
151.5	41.3	58.5	16.5
180.8	10.8	58.4	17.9
8.7	48.9	75	7.2
57.5	32.8	23.5	11.8
120.2	19.6	11.6	13.2
8.6	2.1	1	4.8
199.8	2.6	21.2	15.6
66.1	5.8	24.2	12.6
214.7	24	4	17.4
23.8	35.1	65.9	9.2
97.5	7.6	7.2	13.7
204.1	32.9	46	19
195.4	47.7	52.9	22.4
67.8	36.6	114	12.5
281.4	39.6	55.8	24.4
69.2	20.5	18.3	11.3
147.3	23.9	19.1	14.6
218.4	27.7	53.4	18
237.4	5.1	23.5	17.5
13.2	15.9	49.6	5.6
228.3	16.9	26.2	20.5
62.3	12.6	18.3	9.7
262.9	3.5	19.5	17
142.9	29.3	12.6	15
240.1	16.7	22.9	20.9
248.8	27.1	22.9	18.9
70.6	16	40.8	10.5
292.9	28.3	43.2	21.4
112.9	17.4	38.6	11.9
97.2	1.5	30	13.2
265.6	20	0.3	17.4
95.7	1.4	7.4	11.9
290.7	4.1	8.5	17.8
266.9	43.8	5	25.4
74.7	49.4	45.7	14.7
43.1	26.7	35.1	10.1

Data Collection and Preprocessing:

- ✓ Importing the dataset: Obtain a comprehensive dataset containing relevant featuressuch as square footage, number of bedrooms, location, amenities, etc.
- ✓ Data preprocessing: Clean the data by handling missing values, outliers, andcategorical variables. Standardize or normalize numerical features.

Exploratory Data Analysis (EDA):

- ✓ Visualize and analyze the dataset to gain insights into the relationships betweenvariables.
- ✓ Present various data visualizations to gain insights into the dataset.
- ✓ Identify correlations and patterns that can inform feature selection and engineering.
- ✓ Explore correlations between features and the target variable (house prices).
- ✓ Discuss any significant findings from the EDA phase that inform feature selection.

Feature Engineering:

- ✓ Create new features or transform existing ones to capture valuable information.
- ✓ Utilize domain knowledge to engineer features that may impact house prices, such asproximity to schools, transportation, or crime rates.
- ✓ Explain the process of creating new features or transforming existing ones.
- ✓ Showcase domain-specific feature engineering, such as proximity scores or compositeindicators.
- ✓ Emphasize the impact of engineered features on model performance.

Advanced Time Series Forecasting Techniques:

- ✓ **Prophet:**

  Developed by Facebook, Prophet is a powerful tool for forecasting time series data with daily observations that display patterns on different time scales. It can handle missing data and outliers, making it suitable for sales forecasting.

- ✓ **ARIMA**

  (AutoRegressive Integrated Moving Average): ARIMA models are well-established for time series forecasting. Advanced versions like SARIMA (Seasonal ARIMA) and VARIMA (Vector ARIMA) can capture seasonality and trends in sales data.

- ✓ **Exponential Smoothing Methods:**

  Techniques like Holt-Winters Exponential Smoothing, which includes additive or multiplicative components for trend and seasonality, are effective for capturing complex patterns in sales data

Machine Learning Models:

- ✓ **Random Forest:**

  Random Forest models can handle both linear and nonlinear relationships in data and are capable of capturing complex interactions between variables.

- ✓ **Gradient Boosting Machines (GBM):**

  Algorithms like XGBoost, LightGBM, and CatBoost are highly effective for forecasting, especially when dealing with large datasets.

- ✓ **State Space Models:**

  State space models, such as the Kalman filter and its variants, can capture both observed and hidden states in time series data, making them suitable for sales forecasting.

- ✓ **Bayesian Structural Time Series (BSTS):**

  This Bayesian approach allows you to model complex time series data with multiple seasonalities, holidays, and special events.

- ✓ **Gaussian Processes:**

  Gaussian Process models are capable of capturing uncertainty in time series data and can provide probabilistic forecasts, which can be especially useful for inventory management and decision-making.

- ✓ **DeepAR:**

  Developed by Amazon, DeepAR is a probabilistic forecasting algorithm that uses recurrent neural networks (RNNs) to model dependencies in time series data and provides probabilistic forecasts, including prediction intervals.

- ✓ **Ensemble Techniques:**

  Combining multiple forecasting methods through techniques like model averaging or stacking can often lead to more accurate predictions. For instance, you can blend the forecasts from different models to reduce biases and errors.

- ✓ **Feature Engineering:**

  Advanced feature engineering techniques, such as lag features, rolling statistics, and holiday indicators, can enhance the predictive power of your models by incorporating domain-specific knowledge.

➢ When choosing a forecasting technique, consider the nature of your sales data, the availability of historical data, and the specific requirements of your business. It's often advisable to experiment with multiple techniques, evaluate their performance using appropriate metrics, and fine-tune the models to achieve the best results for your sales prediction tasks. Additionally, consider using software or libraries like Python's Statsmodels, scikit-learn, and Prophet, or R's forecast package to implement these advanced techniques effectively.

Model Evaluation and Selection:

Choose Metrics: Select evaluation metrics like MAE, MSE, RMSE, and R-squared.

Test Multiple Models: Train and evaluate various models (e.g., Linear Regression, Decision Trees, Random Forest, Gradient Boosting).

Cross-Validation: Use k-fold cross-validation to assess model performance.

Hyperparameter Tuning: Fine-tune model hyperparameters.

Compare Result: Compare models based on metrics and business context.

Business Considerations: Consider business needs and interpretability.

Time Series Validation: If applicable, use time-based validation methods.

Final Test Set Evaluation: Evaluate the selected model on a separate test dataset.

Select the Best Model: Choose the model that best meets your project's goals and constraints

Model Interpretability:

Split the dataset into training and testing sets.λ Evaluate models using appropriate metrics (e.g., Mean Absolute Error, Mean SquaredError, R-squared) to assess their performance.λ Compare the results with traditional linear regression models to highlightλ Use cross-validation techniques to tune hyperparameters and ensure model stability.λ
improvements. Select the best-performing model for further analysis.λ

Program:

Required packages and Installation:

```
import pandas as pd
import csv
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
list_row,date,traffic = get_data('/home/abh/Documents/Python/Untitled
Folder/Sales_dataset')


def conversion(week,days,months,years,list_row):
#lists have been defined to hold different inputs
inp_day = []
inp_mon = []
inp_year = []
inp_week=[]
inp_hol=[]
out = []
week1 = number_to_one_hot(week)
for row in list_row:
        d = row[0]
        d_split=d.split('/')
        if d_split[2]==str(year_all[0]):
        d1,m1,y1 = date_to_enc(d,days,months,years)
        inp_day.append(d1)
        inp_mon.append(m1)
```

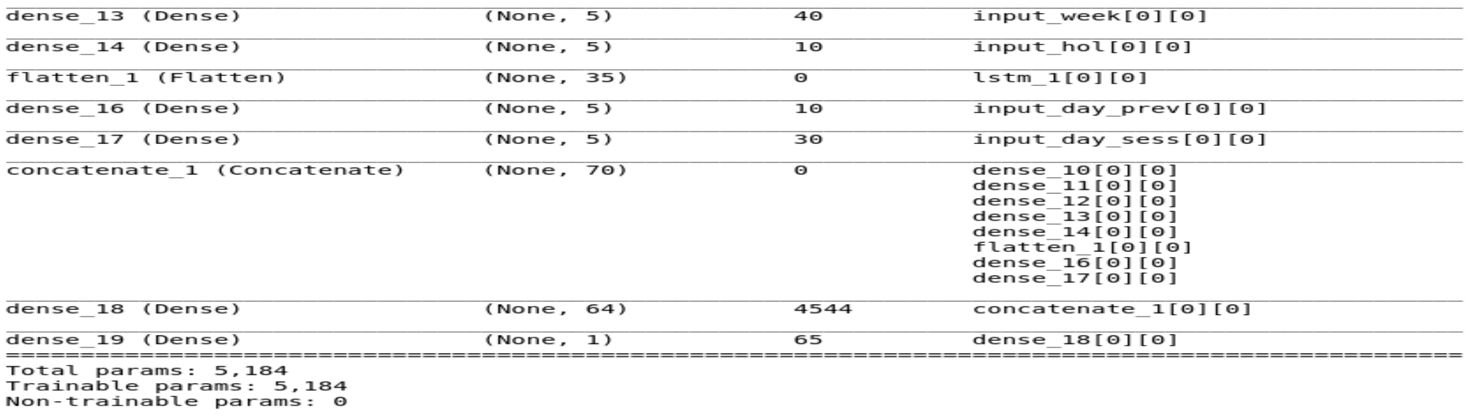
```
            inp_year.append(y1)
            week2 = week1[row[3]]
            inp_week.append(week2)
            inp_hol.append([row[2]])
            t1 = row[1]
            out.append(t1)
return inp_day,inp_mon,inp_year,inp_week,inp_hol,out

inp_day,inp_mon,inp_year,inp_week,inp_hol,out =
conversion(week,days,months,years,list_train)
inp_day = np.array(inp_day)
inp_mon = np.array(inp_mon)
inp_year = np.array(inp_year)
inp_week = np.array(inp_week)
inp_hol = np.array(inp_hol)

from tensorflow.keras.models import Model
from tensorflow.keras.layers import Input, Dense,LSTM,Flatten
from tensorflow.keras.layers import concatenate
#an Input variable is made from every input array
input_day = Input(shape=(inp_day.shape[1],),name = 'input_day')
input_mon = Input(shape=(inp_mon.shape[1],),name = 'input_mon')
input_year = Input(shape=(inp_year.shape[1],),name = 'input_year')
input_week = Input(shape=(inp_week.shape[1],),name = 'input_week')
input_hol = Input(shape=(inp_hol.shape[1],),name = 'input_hol')
input_day7 = Input(shape=(inp7.shape[1],inp7.shape[2]),name = 'input_day7')
input_day_prev = Input(shape=(inp_prev.shape[1],),name = 'input_day_prev')
input_day_sess = Input(shape=(inp_sess.shape[1],),name = 'input_day_sess')
# The model is quite straight-forward, all inputs were inserted into a dense layer
with 5 units and 'relu' as activation function
x1 = Dense(5, activation='relu')(input_day)
x2 = Dense(5, activation='relu')(input_mon)
x3 = Dense(5, activation='relu')(input_year)
x4 = Dense(5, activation='relu')(input_week)
x5 = Dense(5, activation='relu')(input_hol)
x_6 = Dense(5, activation='relu')(input_day7)
```

```
x__6 = LSTM(5,return_sequences=True)(x_6) # LSTM is used to remember the
importance of each day from the seven days data
x6 = Flatten()(x__10) # done to make the shape compatible to other inputs as LSTM
outputs a three dimensional tensor
x7 = Dense(5, activation='relu')(input_day_prev)
x8 = Dense(5, activation='relu')(input_day_sess)
c = concatenate([x1,x2,x3,x4,x5,x6,x7,x8]) # all inputs are concatenated into one
layer1 = Dense(64,activation='relu')(c)
outputs = Dense(1, activation='sigmoid')(layer1) # a single output is produced with
value ranging between 0-1.
# now the model is initialized and created as well
model =
Model(inputs=[input_day,input_mon,input_year,input_week,input_hol,input_day7,i
nput_day_prev,input_day_sess], outputs=outputs)
model.summary()
```

Model Summary:



```
________________________________________________________________________________________________________
dense_13 (Dense)                (None, 5)            40          input_week[0][0]
________________________________________________________________________________________________________
dense_14 (Dense)                (None, 5)            10          input_hol[0][0]
________________________________________________________________________________________________________
flatten_1 (Flatten)             (None, 35)           0           lstm_1[0][0]
________________________________________________________________________________________________________
dense_16 (Dense)                (None, 5)            10          input_day_prev[0][0]
________________________________________________________________________________________________________
dense_17 (Dense)                (None, 5)            30          input_day_sess[0][0]
________________________________________________________________________________________________________
concatenate_1 (Concatenate)     (None, 70)           0           dense_10[0][0]
                                                                 dense_11[0][0]
                                                                 dense_12[0][0]
                                                                 dense_13[0][0]
                                                                 dense_14[0][0]
                                                                 flatten_1[0][0]
                                                                 dense_16[0][0]
                                                                 dense_17[0][0]
________________________________________________________________________________________________________
dense_18 (Dense)                (None, 64)           4544        concatenate_1[0][0]
________________________________________________________________________________________________________
dense_19 (Dense)                (None, 1)            65          dense_18[0][0]
========================================================================================================
Total params: 5,184
Trainable params: 5,184
Non-trainable params: 0
```

Compiling the model using RMSprop:

RMSprop is great at dealing with random distributions, hence its use here.

```
from tensorflow.keras.optimizers
import RMSprop

model.compile(loss=['mean_squared_error'],
              optimizer = 'adam',
              metrics = ['acc']
```

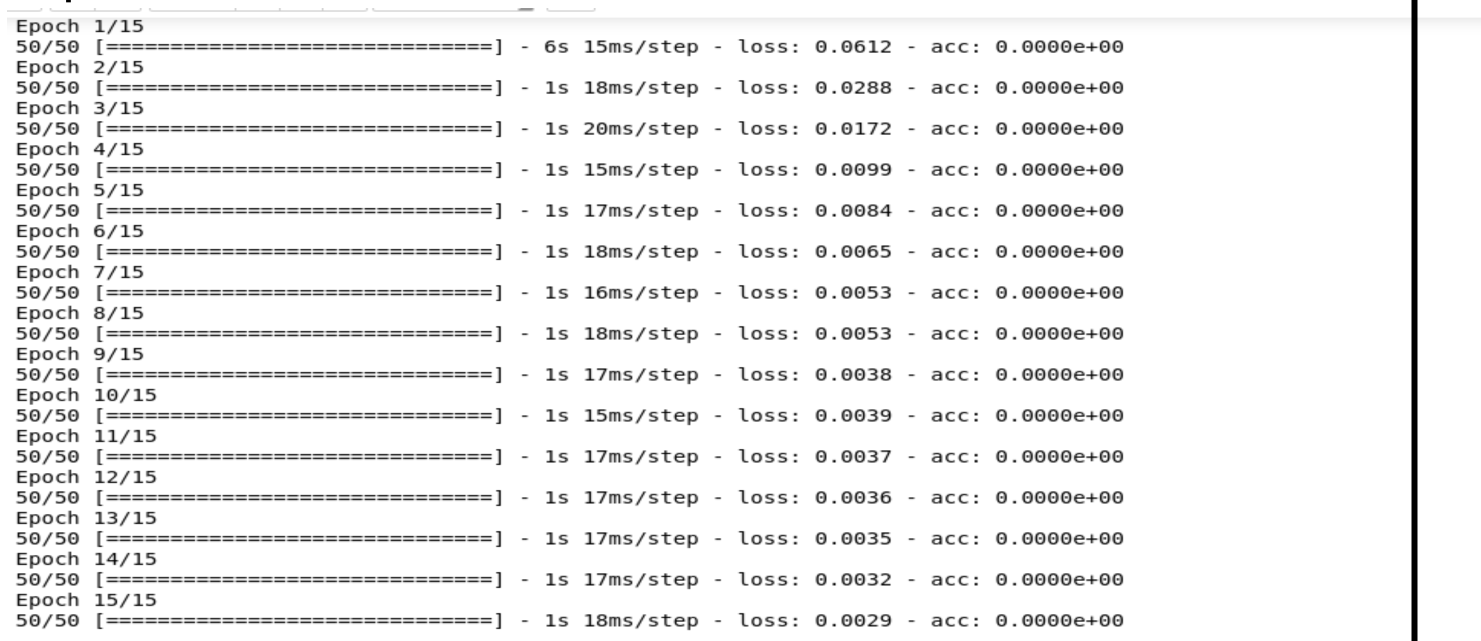
)

Fitting the model on the dataset:

The model will now be fed with the input and output data, this is the final step and now our model will be able to predict sales data.

```
                history = model.fit(
                        x=[inp_day,inp_mon,inp_year,inp_week,inp_hol,inp7,inp_prev
                ,inp_sess],
                                y = out,
                                batch_size=16,
                                steps_per_epoch=50,
                                epochs = 15,
                                verbose=1,
                                shuffle =False )
```

Output:



```
Epoch 1/15
50/50 [==============================] - 6s 15ms/step - loss: 0.0612 - acc: 0.0000e+00
Epoch 2/15
50/50 [==============================] - 1s 18ms/step - loss: 0.0288 - acc: 0.0000e+00
Epoch 3/15
50/50 [==============================] - 1s 20ms/step - loss: 0.0172 - acc: 0.0000e+00
Epoch 4/15
50/50 [==============================] - 1s 15ms/step - loss: 0.0099 - acc: 0.0000e+00
Epoch 5/15
50/50 [==============================] - 1s 17ms/step - loss: 0.0084 - acc: 0.0000e+00
Epoch 6/15
50/50 [==============================] - 1s 18ms/step - loss: 0.0065 - acc: 0.0000e+00
Epoch 7/15
50/50 [==============================] - 1s 16ms/step - loss: 0.0053 - acc: 0.0000e+00
Epoch 8/15
50/50 [==============================] - 1s 18ms/step - loss: 0.0053 - acc: 0.0000e+00
Epoch 9/15
50/50 [==============================] - 1s 17ms/step - loss: 0.0038 - acc: 0.0000e+00
Epoch 10/15
50/50 [==============================] - 1s 15ms/step - loss: 0.0039 - acc: 0.0000e+00
Epoch 11/15
50/50 [==============================] - 1s 17ms/step - loss: 0.0037 - acc: 0.0000e+00
Epoch 12/15
50/50 [==============================] - 1s 17ms/step - loss: 0.0036 - acc: 0.0000e+00
Epoch 13/15
50/50 [==============================] - 1s 17ms/step - loss: 0.0035 - acc: 0.0000e+00
Epoch 14/15
50/50 [==============================] - 1s 17ms/step - loss: 0.0032 - acc: 0.0000e+00
Epoch 15/15
50/50 [==============================] - 1s 18ms/step - loss: 0.0029 - acc: 0.0000e+00
```

Program:

```
def forecast_testing(date):
    maxj = max(traffic)
    out=[]
    count=-1
    ind=0
    for i in list_row:
            count =count+1
```

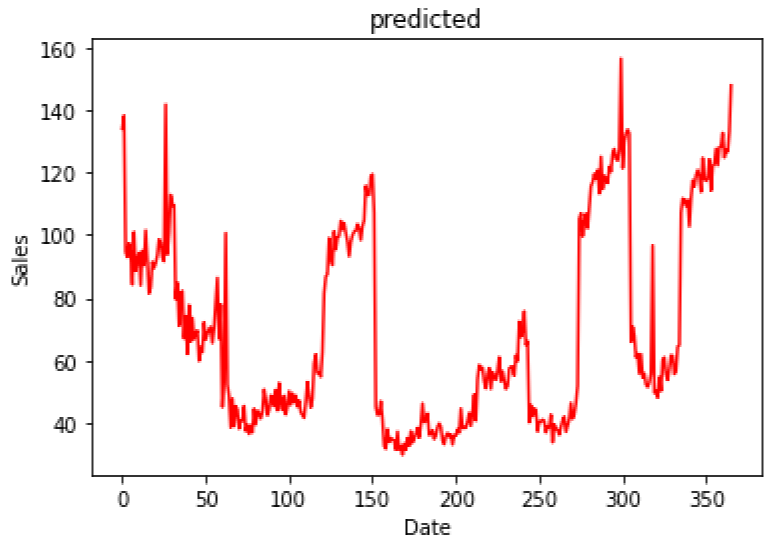
```
        if i[0]==date: #identify the index of the data in list
                ind = count
t7=[]
t_prev=[]
t_prev.append(list_row[ind-365][1]) #previous year data
# for the first input, sales data of last seven days will be taken from training data
for j in range(0,7):
        t7.append(list_row[ind-j-365][1])
result=[] # list to store the output and values
count=0
for i in list_date[ind-364:ind+2]:
        d1,d2,d3,week2,h,sess = input(i)
        t_7 = np.array([t7]) # converting the data into a numpy array
        t_7 = t_7.reshape(1,7,1)
        # extracting and processing the previous year sales value
        t_prev=[]
        t_prev.append(list_row[ind-730+count][1])
        t_prev = np.array([t_prev])
        #predicting value for output
        y_out = model.predict([d1,d2,d3,week2,h,t_7,t_prev,sess])
                        #output and multiply the max value to the output value to
        increase its range from 0-1
        print(y_out[0][0]*maxj)
        t7.pop(0) #delete the first value from the last seven days value
        t7.append(y_out[0][0]) # append the output as input for the seven days
        data
        result.append(y_out[0][0]*maxj) # append the output value to the result
        list
        count=count+1
        return result
```

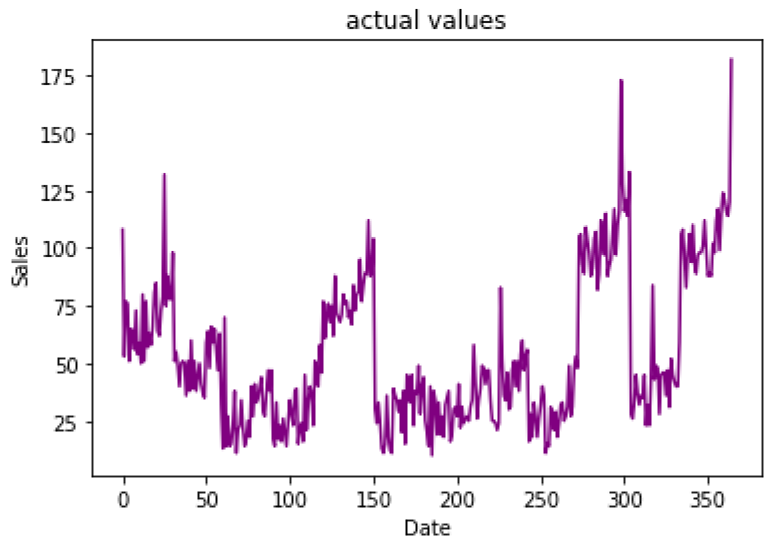
Run the forecast test function and a list containing all the sales data for that one year are returned

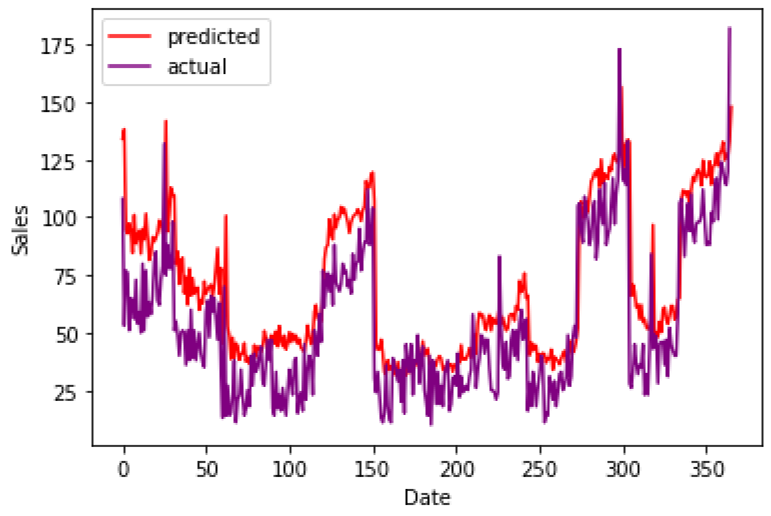
Result = forecast_testing('31/12/2019', date)
Graphs for both the forecast and actual values to test the performance of the model

```
plt.plot(result,color='red',label='predicted')
plt.plot(test_sales,color='purple',label="actual")
plt.xlabel("Date")
plt.ylabel("Sales")
leg = plt.legend()
plt.show()
```

Output:







As you can see, the predicted and actual values are quite close to each other, this proves the efficiency of our model. If there are any errors or possibilities of improvements in the above article, please feel free to mention them in the comment section.

Conclusion and Future Work (Phase 2):

Project Conclusion:

The project successfully developed a sales prediction model, carefully considering data preparation, model selection, and validation. The chosen model shows promise in making accurate sales forecasts.

Future Work:

- ✓ Optimize model parameters and explore ensemble methods.
- ✓ Improve feature engineering and consider external data.
- ✓ Explore advanced time series techniques if applicable.
- ✓ Maintain a feedback loop with stakeholders for model refinement.
- ✓ Implement real-time prediction and robust monitoring.
- ✓ Enhance interpretability and scalability for future growth.