OhioHealth Data Analytics/Modeling Exercise

Submitted by Ali Haider Date 11/02/2020

Importing libraries and the data set

In [291]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import statistics
from statistics import median
from scipy import stats
```

In [223]:

```
def read_csv(file_name):
    data = pd.read_csv(file_name, encoding = 'utf-8')

    # fix encoding related errors
    data.columns = [col.replace('\xa0', '') for col in data.columns]
    for col in data.columns:
        data[col] = data[col].astype(str).str.replace(u'\xa0', '')

    return data
```

In [224]:

```
#Sheet 'student1' of OH_dataset1
df1 = read_csv("student1.csv")
df1.head()
```

Out[224]:

	Studentid	year	score
0	A	94	320
1	A	94	348
2	A	92	365
3	B	92	402
4	B	92	354

In [234]:

```
#Sheet 'student2' of OH_dataset1
df2 = read_csv("student2.csv")
df2.head()
```

Out[234]:

	Studentid	Gender	Age	EconomicStatus
0	A	M	16	high
1	B	F	15	medium
2	C	F	18	high
3	D	M	16	low
4	E	M	16	high

In [235]:

```
#OH_dataset2 of excercise
df3 = read_csv("OH_dataset2.csv")
df3.head()
```

Out[235]:

	Year	Month	TimeSeries
0	2012	1	13
1	2012	2	12
2	2012	3	11
3	2012	4	10
4	2012	5	9

A. Use dataset OH_dataset1.xlsx (note it contains two sheets) and create R/Python code to answer below questions.

1. Write code to retrieve mean and median scores for each student ID?

In [689]:

```
list_dict_scores = []

#Finding list of unique studenID
list_studentID = list(df1["Studentid"])
list_studentID = set(list_studentID)
list_studentID = sorted(list_studentID)

#Make list of scores
for x in list_studentID:
    f =[]
    for i in range(df1.shape[0]):
        if x == list(df1['Studentid'])[i]:
            lst = list(df1['score'])[i]
            f.append(lst)
    r = {x: f}
    list_dict_scores.append(r)


#Make a list of mean and median
list1 = []
for j in range(len(w)):
    letter_dict = list_dict_scores[j]
    values_list = list(letter_dict.values())
    values_list = values_list[0]
    values_list = [int(k) for k in values_list]

    float_mean = statistics.mean(values_list)
    rounded_mean = round(float_mean, 1)

    float_median = median(values_list)

    mean_median_list = [rounded_mean, float_median]
    list1.append(mean_median_list)

#Convert into data frame
Dict_mean_median = dict(zip(list_studentID, list1))
Dict_mean_median = pd.DataFrame.from_dict(Dict_mean_median)
Dict_mean_median.index = ['Mean', 'Median']
Dict_mean_median
```

Out[689]:

	A	B	C	D	E	F	G	H	I
Mean	344.3	367.8	415.3	350.5	494.0	347	422.7	369.2	420.7
Median	348.0	377.0	425.5	350.5	494.0	348	449.0	356.5	411.0

2. Create a new binary variable named result, which indicates whether a student scored above 400?

In [237]:

```
#Print True if student's score is greater than 400 and false when it is not
df1['score'] = pd.to_numeric(df1['score'])
df = df1["score"] >= 400
df = pd.concat([df1['Studentid'], df], axis=1)
df
```

Out[237]:

	Studentid	score
0	A	False
1	A	False
2	A	False
3	B	True
4	B	False
5	B	False
6	B	True
7	C	True
8	C	True
9	C	True
10	C	False
11	C	True
12	C	False
13	D	False
14	D	False
15	E	True
16	E	True
17	F	False
18	F	False
19	F	False
20	G	False
21	G	True
22	G	True
23	H	False
24	H	True
25	H	False
26	H	False
27	I	True

	Studentid	score
27	I	True
28	I	True
29	I	True

3. Write code to find out the gender and age of each student ID?

In [401]:

```
pd.concat([df2['Studentid'], df2['Gender'], df2['Age']], axis=1, keys=['Studentid', 'Gender', 'Age'])
```

Out[401]:

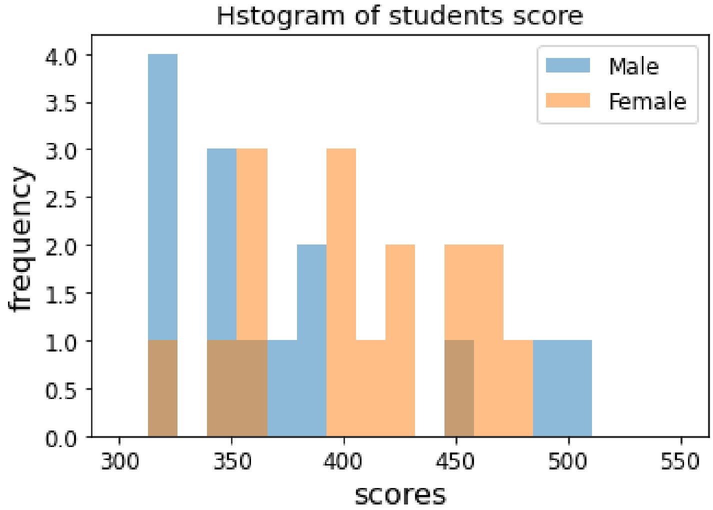
	Studentid	Gender	Age
0	A	M	16
1	B	F	15
2	C	F	18
3	D	M	16
4	E	M	16
5	F	M	15
6	G	F	16
7	H	M	16
8	I	F	17

4. Create a histogram to find out the distribution of score by gender? Is the distribution normal?

In [671]:

```
#Making list of male and female scores
Male_scores = []
Female_scores = []

for i in range(len(df1)):

    student_id_1 = list(df1['Studentid'])[i]

    for j in range(len(df2)):

        student_id_2 = list(df2['Studentid'])[j]
        student_gender_2 = list(df2['Gender'])[j]

        if  student_id_1 == student_id_2 and student_gender_2 == 'M':
            Male_score = list(df1['score'])[i]
            Male_scores.append(Male_score)

        elif student_id_1 == student_id_2 and student_gender_2 == 'F':
            Female_score = list(df1['score'])[i]
            Female_scores.append(Female_score)


#Plotting the histogram
x = Male_scores
y = Female_scores
```

```
bins = np.linspace(300, 550, 20)

plt.title("Hstogram of students score")
plt.hist(x, bins, alpha=0.5, label='Male')
plt.hist(y, bins, alpha=0.5, label='Female')
plt.legend(loc='upper right')
plt.xlabel('scores', fontsize=16)
plt.ylabel('frequency', fontsize=16)
plt.show()
```



Distribution of Male scores is not normally distributed while female scores can be considered normally distributed due to the bell shape of the histogram.

5. Is there any statistical difference between male and female scores?

In [292]:

```
#Statistical significance testing
t_check=stats.ttest_ind(Male_scores, Female_scores)
t_check
alpha=0.05
if(t_check[1]<alpha):
    print('Male and Female scores are statisticaly different')
else:
    print('The difference in Male and Female scores is not statisticaly significant')
```

```
The difference in Male and Female scores is not statisticaly significant
```

6.For each student ID, what is the most recent score and which year was it?

In [373]:

```
#most recent score and year
df1['year'] = df1['year'].astype(int)

inDex = []
Recent_year_list = []

for j in range(len(df2)):
    recent_year = 0
    for i in range(len(df1)):

            student_id_1 = list(df1['Studentid'])[i]
```

```
            student_id_2 = list(df2['Studentid'])[j]
            student_year = list(df1['year'])[i]

            if  student_id_1 == student_id_2:

                if recent_year < student_year:
                    recent_score = list(df1['score'])[i]
                    recent_year = max(recent_year, student_year)
                    index = i

    inDex.append(index)
    Recent_year_list.append(recent_year)


recent_score = []
for k in range(len(inDex)):
    recent_score.append(list(df1['score'])[inDex[k]])


student_recent_score_year = {'Studentid': list(df2['Studentid']), 'Recent_score': recent_
score, \
                             'Recent_year': Recent_year_list}
df5 = pd.DataFrame(data=student_recent_score_year)
df5
```

Out[373]:

	Studentid	Recent_score	Recent_year
0	A	320	94
1	B	402	92
2	C	480	94
3	D	380	95
4	E	499	94
5	F	379	95
6	G	462	94
7	H	450	95
8	I	401	92

B. Use dataset OH_dataset1.xlsx (note it contains two sheets) and create R/Python code to answer below questions.

1. How would you recode gender as a 0/1 binary indicator?

In [388]:

```
gender_bin = pd.get_dummies(df2.Gender, prefix='Gender')
gender_bin.drop(['Gender_M'], axis=1)

pd.concat([df2['Studentid'], df2['Gender'], gender_bin['Gender_F']], axis=1, keys=['Stud
entid', 'Gender', 'Gender_F'])
```

Out[388]:

	Studentid	Gender	Gender_F
0	A	M	0
1	B	F	1
2	C	F	1

	Studentid	Gender	Gender_F
3	D	M	0
4	E	M	0
5	F	M	0
6	G	F	1
7	H	M	0
8	I	F	1

Its 0 for male and 1 for female

2. Recode age as an ordinal variable using the following criteria:

a. < 15 = 1

b. 15 – 18 (both inclusive) = 2

c. > 18 = 3

In [404]:

```
Ordinal_variable_age = []
age_list = [int(i) for i in list(df2['Age'])]

for i in range(len(df2['Age'])):

    if age_list[i] < 15:
        Ordinal_variable_age.append(1)

    elif age_list[i] >= 15 and age_list[i] <= 18:
        Ordinal_variable_age.append(2)
    else:
        Ordinal_variable_age.append(3)

categorical_age = {'Age group': Ordinal_variable_age}
categorical_age = pd.DataFrame(data= categorical_age)
categorical_age


pd.concat([df2['Studentid'], df2['Age'], categorical_age['Age group']], axis=1, keys=['S
tudentid', 'Age', 'Age group'])
```

Out[404]:

	Studentid	Age	Age group
0	A	16	2
1	B	15	2
2	C	18	2
3	D	16	2
4	E	16	2
5	F	15	2
6	G	16	2
7	H	16	2
8	I	17	2

3. Build a machine learning algorithm to predict student’s score using the other features. Use any model specification and data transformation you think is appropriate.

In [458]:

```
#Making the data structure

list_year = []
list_score = []
list_Gender = []
list_Age = []
list_economic = []

for i in range(len(df1)):
    for j in range(len(df2)):

        if list(df1['Studentid'])[i] == list(df2['Studentid'])[j]:
            list_year.append(list(df1['year'])[i])
            list_score.append(list(df1['score'])[i])
            list_Gender.append(list(df2['Gender'])[j])
            list_Age.append(list(df2['Age'])[j])
            list_economic.append(list(df2['EconomicStatus'])[j])

data_dict = {'Studentid': list(df1['Studentid']), 'year':  list_year, \
             'EconomicStatus': list_economic, 'Gender': list_Gender, 'Age': list_Age,  '
score': list_score}


df6 = pd.DataFrame(data = data_dict)
df6.head()
```

Out[458]:

	Studentid	year	EconomicStatus	Gender	Age	score
0	A	94	high	M	16	320
1	A	94	high	M	16	348
2	A	92	high	M	16	365
3	B	92	medium	F	15	402
4	B	92	medium	F	15	354

Since the details in questions are not given so I made two assumptions which are as follow-

- **We have to predict scores of a new student who is not in the given list**
- **The scores of each student are not the repition of a same course exam.**

we can drop studentid column as it is not important for new student's score prediction

In [424]:

```
df6 = df6.drop(['Studentid'], axis=1)
```

Now we are all set and have to convert categorical columns into dummy variables.

In [438]:

```
#Making dummy variables of economic status columns
economic_bin = pd.get_dummies(df6.EconomicStatus, prefix='EconomicStatus')

#dropping medium column to prevent dummy variable trap
economic_bin = economic_bin[['EconomicStatus_high', 'EconomicStatus_low'] ]

#Making dummy variables of gender columns
```

```python
gender_bin = pd.get_dummies(df6.Gender, prefix='Gender')

#dropping female column to prevent dummy variable trap
gender_bin = gender_bin[['Gender_M']]

#Concatenate all the columns
Data = pd.concat([df6['year'], economic_bin, gender_bin, df6['Age'], df6['score']], axis
=1)
Data.head()
```

Out[438]:

	year	EconomicStatus_high	EconomicStatus_low	Gender_M	Age	score
0	94	1	0	1	16	320
1	94	1	0	1	16	348
2	92	1	0	1	16	365
3	92	0	0	0	15	402
4	92	0	0	0	15	354

In [461]:

```python
# Random Forest Regression

X = Data.iloc[:, 1:-1].values
y = Data.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, shuffle=True
, \
                                                    random_state=4)

# Feature Scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestRegressor
regressor = RandomForestRegressor(n_estimators = 5, random_state = 0)
regressor.fit(X_train, y_train)

#Predicting the Test set results
y_pred = regressor.predict(X_test)
y_pred_train = regressor.predict(X_train)

#Evaluating the Algorithm
from sklearn import metrics
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 47.01613095238095
Mean Squared Error: 3832.81524606009
Root Mean Squared Error: 61.909734663137506
```

In [678]:

```python
# Support Vector Machine
from sklearn.svm import SVC
regressor = SVC(kernel = 'linear', random_state =0)
regressor.fit(X_train, y_train)

#Predicting the Test set results
y_pred = regressor.predict(X_test)
y_pred_train = regressor.predict(X_train)
```

```
#Evaluating the Algorithm
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 0.25
Mean Squared Error: 0.25
Root Mean Squared Error: 0.5
```

The root mean squared error of random forest is less than that of SVM so we will chose random forest as our model.

The mean absolute error is 12% of the avergae score value. It has done a reasonably good job on our small dataset.

4. Create a new binary variable named result, which indicates whether a student scored above 400. Build a machine learning algorithm using this new binary variable as response variable and other features as predictors. Use any model specification and data transformation you think is appropriate

In [484]:

```
#Print 1 if student's score is greater than 400 and 0 when it is not
df1['score'] = pd.to_numeric(df1['score'])
df = df1["score"] > 400

binary_score = []
for i in range(len(df)):
    if df[i] == True:
       binary_score.append(1)
    else:
       binary_score.append(0)

binary_score = {'Binary_score': binary_score}
binary_score = pd.DataFrame(data = binary_score)

Data = pd.concat([df6['year'], economic_bin, gender_bin, df6['Age'], binary_score], axis
=1)
Data.head()
```

Out[484]:

	year	EconomicStatus_high	EconomicStatus_low	Gender_M	Age	Binary_score
0	94	1	0	1	16	0
1	94	1	0	1	16	0
2	92	1	0	1	16	0
3	92	0	0	0	15	1
4	92	0	0	0	15	0

In [679]:

```
# Random Forest classification

X = Data.iloc[:, 1:-1].values
y = Data.iloc[:, -1].values

# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, shuffle=True
, \
                                                    random_state=4)
```

```
# Feature Scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

# Fitting Random Forest Classification to the Training set
from sklearn.ensemble import RandomForestClassifier
classifier = RandomForestClassifier(n_estimators = 5, criterion = 'entropy', random_stat
e = 0)
classifier.fit(X_train, y_train)


#Predicting the Test set results
y_pred = classifier.predict(X_test)
y_pred_train = classifier.predict(X_train)

#Evaluating the Algorithm
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score

print("Confusion matrix is")
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print("Accuracy from random forest model is" ,accuracy_score(y_test, y_pred))
```

```
Confusion matrix is
[[2 1]
 [1 4]]
              precision    recall  f1-score   support

           0       0.67      0.67      0.67         3
           1       0.80      0.80      0.80         5

    accuracy                           0.75         8
   macro avg       0.73      0.73      0.73         8
weighted avg       0.75      0.75      0.75         8

Accuracy from random forest model is 0.75
```

In [680]:

```
# Training the SVM model on the Training set
from sklearn.svm import SVC
classifier = SVC(kernel = 'linear', random_state = 0)
classifier.fit(X_train, y_train)


#Predicting the Test set results
y_pred = classifier.predict(X_test)
y_pred_train = classifier.predict(X_train)

#Evaluating the Algorithm
print("Confusion matrix is")
print(confusion_matrix(y_test,y_pred))
print(classification_report(y_test,y_pred))
print("Accuracy from random forest model is" ,accuracy_score(y_test, y_pred))
```

```
Confusion matrix is
[[3 0]
 [2 3]]
              precision    recall  f1-score   support

           0       0.60      1.00      0.75         3
           1       1.00      0.60      0.75         5

    accuracy                           0.75         8
   macro avg       0.80      0.80      0.75         8
weighted avg       0.85      0.75      0.75         8

Accuracy from random forest model is 0.75
```

Both Random forest and Support vector machine showed 75% accuracy on 25% test set.

C. Use dataset, OH_dataset2.xlsx (note it's a monthly time series dataset) and create R/Python code to answer below questions.

1. Trend the series using a visualization plot.

In [520]:

```
df3.head()
```

Out[520]:

	Year	Month	TimeSeries
0	2012	1	13
1	2012	2	12
2	2012	3	11
3	2012	4	10
4	2012	5	9

In [539]:

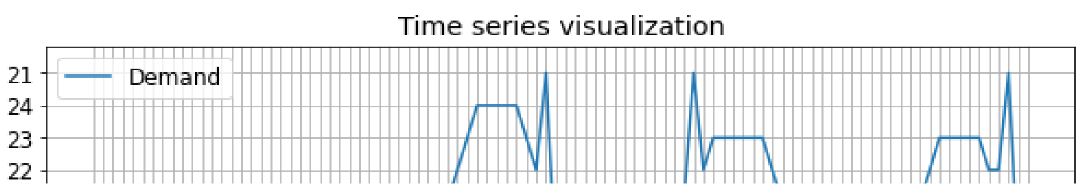
```
# Concatinating Year and Month column
Data_Time_series = df3['Year'] + '-' + df3['Month']
Data_Time_series = pd.DataFrame({'YearMonth':Data_Time_series})

Data_Time_series = pd.concat([Data_Time_series, df3['TimeSeries']], axis=1)
Data_Time_series.head()
```

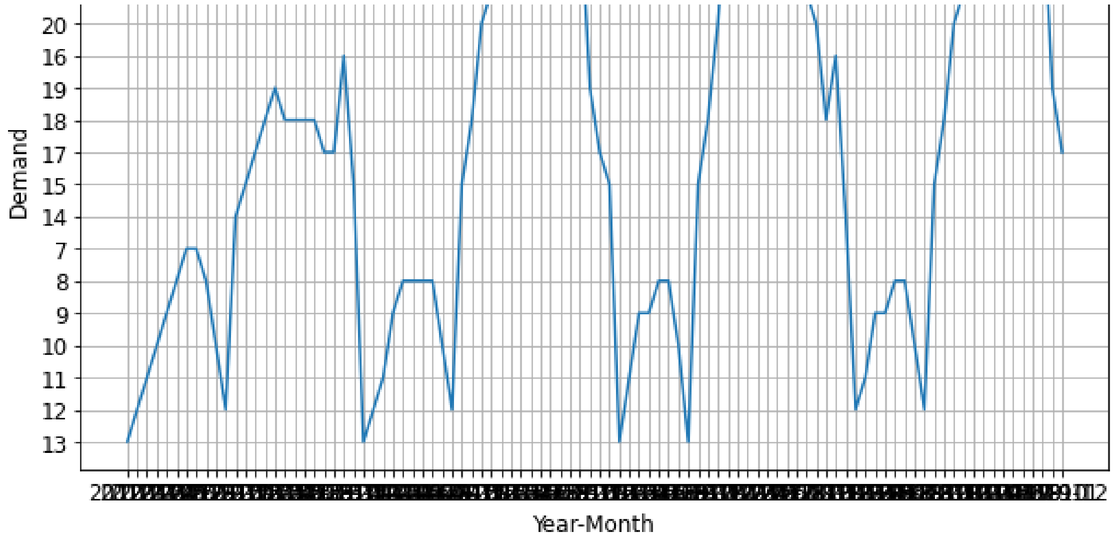
Out[539]:

	YearMonth	TimeSeries
0	2012-1	13
1	2012-2	12
2	2012-3	11
3	2012-4	10
4	2012-5	9

In [540]:

```
plt.rc('font', size=12)
fig, ax = plt.subplots(figsize=(10, 6))

# Specify how our lines should look
ax.plot(Data_Time_series.YearMonth, Data_Time_series.TimeSeries, color='tab:blue', label
='Demand')

# Same as above
ax.set_xlabel('Year-Month')
ax.set_ylabel('Demand')
ax.set_title('Time series visualization')
ax.grid(True)
ax.legend(loc='upper left');
```

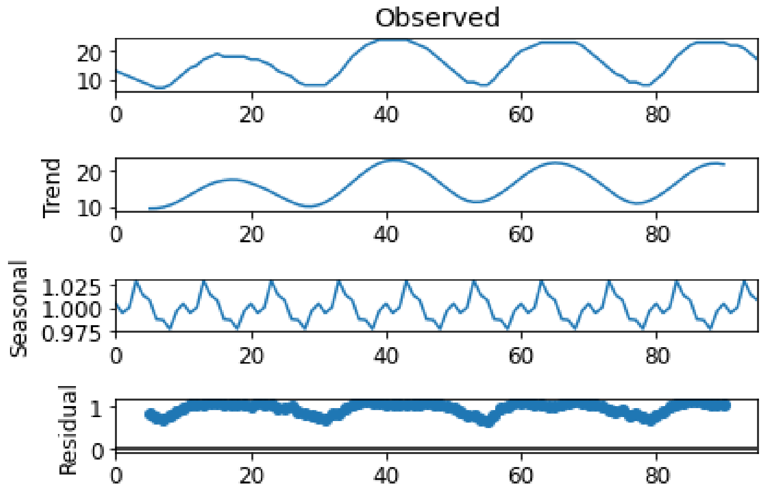




2. Filter the level, trend and seasonality from the series

In [588]:

```
# Time Series Decomposition
from statsmodels.tsa.seasonal import seasonal_decompose
df8 = Data_Time_series['TimeSeries']
df8 = df8.values
result = seasonal_decompose(df8, model='multiplicative', period=10)
result.plot()
pyplot.show()

df8 = df8.astype(np.float)
print("Level of the series or average is ", df8.mean())
```




```
Level of the series or average is  16.0
```

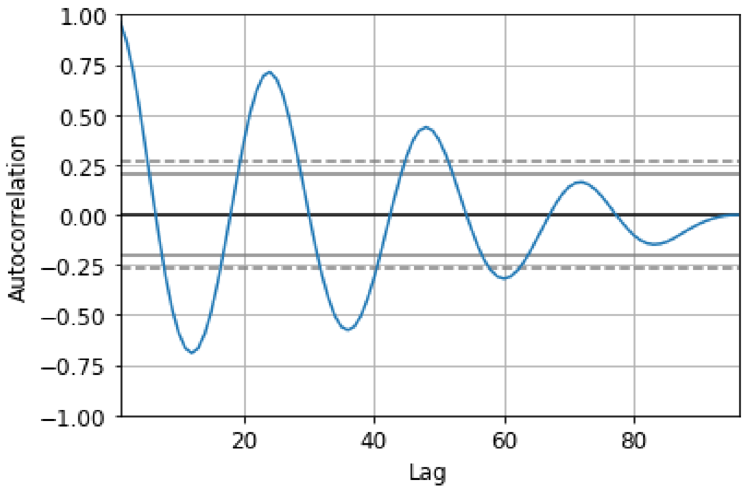
3. Use the dataset and any time series model building technique to predict the monthly value for next the 12 months for 2020

In [681]:

```
# Autocorellation plot
from pandas import datetime
from pandas.plotting import autocorrelation_plot
```

```
series = pd.Series(df8)
autocorrelation_plot(series)
plt.show()

print('Hence we will chose a lag of 3')
```

```
<ipython-input-681-fe3b82c3b118>:2: FutureWarning: The pandas.datetime class is deprecate
d and will be removed from pandas in a future version. Import from datetime module instea
d.
  from pandas import datetime
```

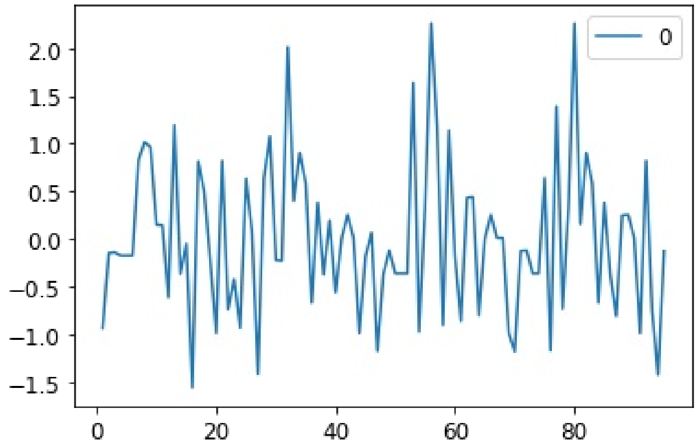


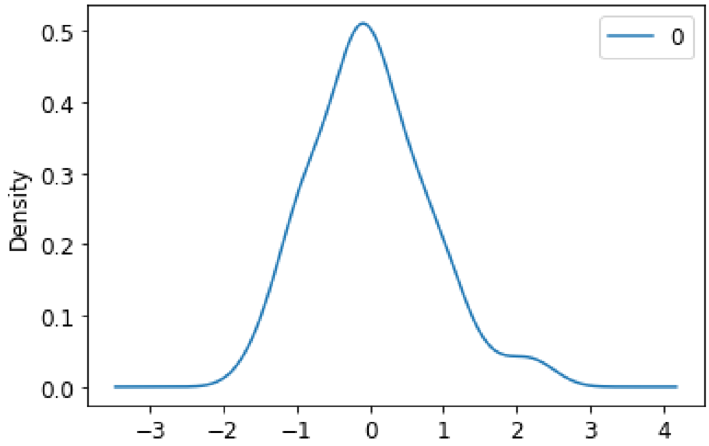

```
Hence we will chose a lag of 3
```

In [682]:

```
# ARIMA Forecasting
from statsmodels.tsa.arima_model import ARIMA
from pandas import DataFrame

# fit model
model = ARIMA(series, order=(3,1,0))
model_fit = model.fit(disp=0)
print(model_fit.summary())
# plot residual errors
residuals = DataFrame(model_fit.resid)
residuals.plot()
plt.show()
residuals.plot(kind='kde')
plt.show()
print(residuals.describe())
```

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:                    D.y   No. Observations:                   95
Model:                 ARIMA(3, 1, 0)   Log Likelihood                -112.358
Method:                       css-mle   S.D. of innovations              0.784
Date:                Mon, 02 Nov 2020   AIC                            234.716
Time:                        19:31:07   BIC                            247.485
Sample:                             1   HQIC                           239.876

==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const         -0.0651      0.420     -0.155      0.877      -0.889       0.759
ar.L1.D.y      0.8067      0.098      8.202      0.000       0.614       0.999
ar.L2.D.y      0.2481      0.126      1.962      0.050       0.000       0.496
ar.L3.D.y     -0.2415      0.099     -2.428      0.015      -0.436      -0.047
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1           -1.8929           -0.0000j            1.8929           -0.5000
AR.2            1.4600           -0.2359j            1.4790           -0.0255
AR.3            1.4600           +0.2359j            1.4790            0.0255
-----------------------------------------------------------------------------
```





```
              0
count  95.000000
mean    0.004005
std     0.792183
min    -1.559591
25%    -0.491131
50%    -0.119798
75%     0.437128
max     2.260235
```

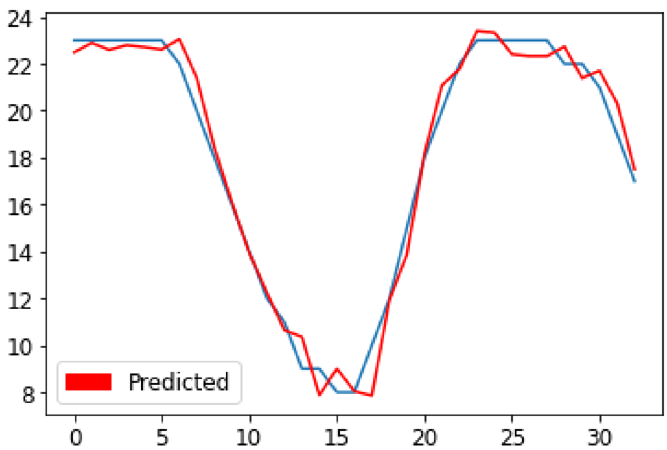
Its good to understand the data while applying a model. Fom the graph we can see that the Errors are not centered. The distribution of the residual errors is displayed.The results show that indeed there is a slight bias in the prediction (a non-zero mean in the residuals).

In [683]:

```
from sklearn.metrics import mean_squared_error
import matplotlib.patches as mpatches

X = series.values
size = int(len(X) * 0.66)
train, test = X[0:size], X[size:len(X)]
history = [x for x in train]
predictions = list()
for t in range(len(test)):
    model = ARIMA(history, order=(5,1,0))
    model_fit = model.fit(disp=0)
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    jobs = test[t]
    history.append(jobs)
    print('predicted=%f, expected=%f' % (yhat, jobs))
error = mean_squared_error(test, predictions)
print('Test MSE: %.3f' % error)
# plot
red_patch = mpatches.Patch(color='red', label='Predicted')
plt.legend(handles=[red_patch])
plt.plot(test)
```

```
plt.plot(predictions, color='red')
plt.show()
```

```
predicted=22.494288, expected=23.000000
predicted=22.888132, expected=23.000000
predicted=22.597231, expected=23.000000
predicted=22.796899, expected=23.000000
predicted=22.710947, expected=23.000000
predicted=22.611069, expected=23.000000
predicted=23.057288, expected=22.000000
predicted=21.401094, expected=20.000000
predicted=18.434682, expected=18.000000
predicted=16.108357, expected=16.000000
predicted=13.936787, expected=14.000000
predicted=12.249132, expected=12.000000
predicted=10.648817, expected=11.000000
predicted=10.355641, expected=9.000000
predicted=7.869727, expected=9.000000
predicted=8.998038, expected=8.000000
predicted=8.040724, expected=8.000000
predicted=7.852205, expected=10.000000
predicted=11.932803, expected=12.000000
predicted=13.865628, expected=15.000000
predicted=18.164987, expected=18.000000
predicted=21.085890, expected=20.000000
predicted=21.775244, expected=22.000000
predicted=23.401709, expected=23.000000
predicted=23.332083, expected=23.000000
predicted=22.410572, expected=23.000000
predicted=22.331390, expected=23.000000
predicted=22.331221, expected=23.000000
predicted=22.733753, expected=22.000000
predicted=21.395724, expected=22.000000
predicted=21.703453, expected=21.000000
predicted=20.334096, expected=19.000000
predicted=17.504334, expected=17.000000
Test MSE: 0.618
```



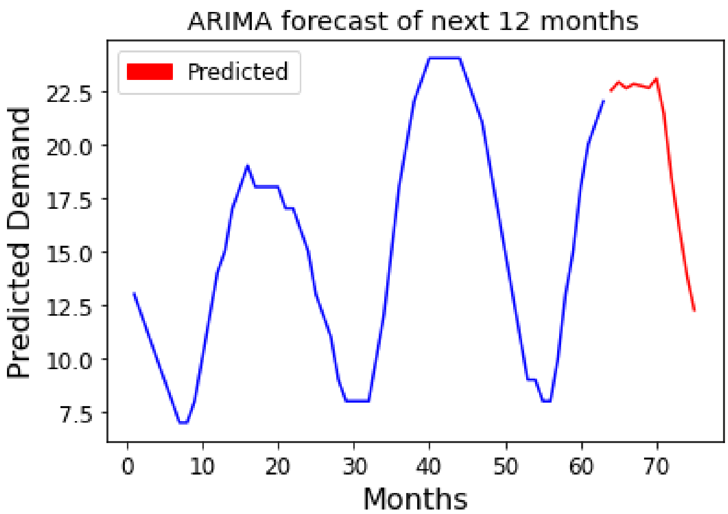
Comparing test results on cross validation set allow us to validate the model. From the above analysis it looks like a good fit so we should move forward to predict for next 12 months

In [684]:

```
#Making predicitions of next 12 months
history = [x for x in train]
predictions = list()
for t in range(12):
    model = ARIMA(history, order=(5,1,0))
    model_fit = model.fit(disp=0)
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    jobs = test[t]
    history.append(jobs)
```

```
x_train = []
for i in range(len(train)):
    x_train.append(i+1)

length_train = len(train)
length_predictions = len(predictions)
length_forecast = length_train + length_predictions
x_predictions =[]

for i in range(length_train, length_forecast):
    x_predictions.append(i+1)

print(x_predictions)
plt.plot(x_train, train, color='blue')
plt.plot(x_predictions, predictions, color='red')


plt.title("ARIMA forecast of next 12 months")
plt.xlabel('Months', fontsize=16)
plt.ylabel('Predicted Demand', fontsize=16)
red_patch = mpatches.Patch(color='red', label='Predicted')
plt.legend(handles=[red_patch])
plt.show()
```

```
[64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75]
```



Now using LSTM for time series

In [787]:

```
#LSTM for time series

test1 = test
train1 = train.reshape(-1, 1)

# Feature Scaling
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(train1)
```

In [788]:

```
# Creating a data structure with 20 timesteps and 1 output
X_train = []
y_train = []
for i in range(20, len(train1)):
    X_train.append(training_set_scaled[i-20:i, 0])
```

```
    y_train.append(training_set_scaled[i, 0])
X_train, y_train = np.array(X_train), np.array(y_train)
```

In [789]:

```
# Reshaping
X_train = np.reshape(X_train, (X_train.shape[0], X_train.shape[1], 1))
```

In [790]:

```
#Building the RNN

# Importing the Keras libraries and packages
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Dropout

# Initialising the RNN
regressor = Sequential()

# Adding the first LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True, input_shape = (X_train.shape[1],
1)))
regressor.add(Dropout(0.2))

# Adding a second LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a third LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50, return_sequences = True))
regressor.add(Dropout(0.2))

# Adding a fourth LSTM layer and some Dropout regularisation
regressor.add(LSTM(units = 50))
regressor.add(Dropout(0.2))

# Adding the output layer
regressor.add(Dense(units = 1))

# Compiling the RNN
regressor.compile(optimizer = 'adam', loss = 'mean_squared_error')

# Fitting the RNN to the Training set
regressor.fit(X_train, y_train, epochs = 500, batch_size = 5)
```

```
Epoch 1/500
9/9 [==============================] - 0s 32ms/step - loss: 0.2200
Epoch 2/500
9/9 [==============================] - 0s 22ms/step - loss: 0.1723
Epoch 3/500
9/9 [==============================] - 0s 36ms/step - loss: 0.1407
Epoch 4/500
9/9 [==============================] - 0s 50ms/step - loss: 0.1365
Epoch 5/500
9/9 [==============================] - 0s 31ms/step - loss: 0.1273
Epoch 6/500
9/9 [==============================] - 0s 29ms/step - loss: 0.1154
Epoch 7/500
9/9 [==============================] - 0s 32ms/step - loss: 0.1252
Epoch 8/500
9/9 [==============================] - 0s 37ms/step - loss: 0.1097
Epoch 9/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0796
Epoch 10/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0747
Epoch 11/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0873
Epoch 12/500
9/9 [==============================] - 0s 29ms/step - loss: 0.0823
Epoch 13/500
```

```
9/9 [==============================] - 0s 38ms/step - loss: 0.0788
Epoch 14/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0776
Epoch 15/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0659
Epoch 16/500
9/9 [==============================] - 0s 31ms/step - loss: 0.0839
Epoch 17/500
9/9 [==============================] - 0s 29ms/step - loss: 0.0752
Epoch 18/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0748
Epoch 19/500
9/9 [==============================] - 0s 45ms/step - loss: 0.0573
Epoch 20/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0587
Epoch 21/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0629
Epoch 22/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0487
Epoch 23/500
9/9 [==============================] - 0s 32ms/step - loss: 0.0345
Epoch 24/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0368
Epoch 25/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0346
Epoch 26/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0265
Epoch 27/500
9/9 [==============================] - 0s 32ms/step - loss: 0.0284
Epoch 28/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0236
Epoch 29/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0230
Epoch 30/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0262
Epoch 31/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0195
Epoch 32/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0133
Epoch 33/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0211
Epoch 34/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0174
Epoch 35/500
9/9 [==============================] - 0s 29ms/step - loss: 0.0234
Epoch 36/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0119
Epoch 37/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0156
Epoch 38/500
9/9 [==============================] - 0s 30ms/step - loss: 0.0150
Epoch 39/500
9/9 [==============================] - 0s 31ms/step - loss: 0.0161
Epoch 40/500
9/9 [==============================] - 0s 30ms/step - loss: 0.0190
Epoch 41/500
9/9 [==============================] - 0s 29ms/step - loss: 0.0282
Epoch 42/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0306
Epoch 43/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0192
Epoch 44/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0149
Epoch 45/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0110
Epoch 46/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0078
Epoch 47/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0159
Epoch 48/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0097
Epoch 49/500
```

```
9/9 [==============================] - 0s 30ms/step - loss: 0.0107
Epoch 50/500
9/9 [==============================] - 0s 29ms/step - loss: 0.0187
Epoch 51/500
9/9 [==============================] - 0s 30ms/step - loss: 0.0089
Epoch 52/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0169
Epoch 53/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0093
Epoch 54/500
9/9 [==============================] - 0s 30ms/step - loss: 0.0129
Epoch 55/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0092
Epoch 56/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0097
Epoch 57/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0141
Epoch 58/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0170
Epoch 59/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0174
Epoch 60/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0116
Epoch 61/500
9/9 [==============================] - 0s 43ms/step - loss: 0.0142
Epoch 62/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0092
Epoch 63/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0115
Epoch 64/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0082
Epoch 65/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0083
Epoch 66/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0095
Epoch 67/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0130
Epoch 68/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0101
Epoch 69/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0107: 0s - loss: 0.
Epoch 70/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0091
Epoch 71/500
9/9 [==============================] - 0s 31ms/step - loss: 0.0140
Epoch 72/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0135
Epoch 73/500
9/9 [==============================] - 0s 26ms/step - loss: 0.0097
Epoch 74/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0113
Epoch 75/500
9/9 [==============================] - 0s 29ms/step - loss: 0.0152
Epoch 76/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0124
Epoch 77/500
9/9 [==============================] - 0s 29ms/step - loss: 0.0065
Epoch 78/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0138
Epoch 79/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0117
Epoch 80/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0233
Epoch 81/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0166
Epoch 82/500
9/9 [==============================] - 0s 31ms/step - loss: 0.0202
Epoch 83/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0190
Epoch 84/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0177
Epoch 85/500
```

```
9/9 [==============================] - 0s 39ms/step - loss: 0.0101
Epoch 86/500
9/9 [==============================] - 0s 44ms/step - loss: 0.0082
Epoch 87/500
9/9 [==============================] - 0s 29ms/step - loss: 0.0103
Epoch 88/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0109
Epoch 89/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0089
Epoch 90/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0070
Epoch 91/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0068
Epoch 92/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0084
Epoch 93/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0103
Epoch 94/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0107
Epoch 95/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0071
Epoch 96/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0137
Epoch 97/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0085
Epoch 98/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0099
Epoch 99/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0058
Epoch 100/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0099
Epoch 101/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0064
Epoch 102/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0087
Epoch 103/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0062
Epoch 104/500
9/9 [==============================] - 0s 44ms/step - loss: 0.0083
Epoch 105/500
9/9 [==============================] - 0s 43ms/step - loss: 0.0096
Epoch 106/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0089
Epoch 107/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0081
Epoch 108/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0093
Epoch 109/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0063
Epoch 110/500
9/9 [==============================] - 0s 46ms/step - loss: 0.0049
Epoch 111/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0079
Epoch 112/500
9/9 [==============================] - 0s 45ms/step - loss: 0.0092
Epoch 113/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0082
Epoch 114/500
9/9 [==============================] - 0s 26ms/step - loss: 0.0074
Epoch 115/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0033
Epoch 116/500
9/9 [==============================] - 0s 25ms/step - loss: 0.0070
Epoch 117/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0076: 0s - loss: 0.007
Epoch 118/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0128
Epoch 119/500
9/9 [==============================] - 0s 26ms/step - loss: 0.0067
Epoch 120/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0064
Epoch 121/500
```

```
9/9 [==============================] - 0s 28ms/step - loss: 0.0054
Epoch 122/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0038
Epoch 123/500
9/9 [==============================] - 0s 42ms/step - loss: 0.0066
Epoch 124/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0071
Epoch 125/500
9/9 [==============================] - 0s 46ms/step - loss: 0.0048
Epoch 126/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0051
Epoch 127/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0075
Epoch 128/500
9/9 [==============================] - 0s 32ms/step - loss: 0.0090
Epoch 129/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0062
Epoch 130/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0054
Epoch 131/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0076
Epoch 132/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0061
Epoch 133/500
9/9 [==============================] - 0s 43ms/step - loss: 0.0045
Epoch 134/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0054
Epoch 135/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0065
Epoch 136/500
9/9 [==============================] - 0s 42ms/step - loss: 0.0071
Epoch 137/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0057
Epoch 138/500
9/9 [==============================] - 0s 42ms/step - loss: 0.0046
Epoch 139/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0046
Epoch 140/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0049
Epoch 141/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0071
Epoch 142/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0063
Epoch 143/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0049
Epoch 144/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0059
Epoch 145/500
9/9 [==============================] - 0s 44ms/step - loss: 0.0056
Epoch 146/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0087
Epoch 147/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0059
Epoch 148/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0052
Epoch 149/500
9/9 [==============================] - 0s 44ms/step - loss: 0.0044
Epoch 150/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0056
Epoch 151/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0070
Epoch 152/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0029
Epoch 153/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0055
Epoch 154/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0053
Epoch 155/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0063
Epoch 156/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0039
Epoch 157/500
```

```
9/9 [==============================] - 0s 26ms/step - loss: 0.0051
Epoch 158/500
9/9 [==============================] - 0s 29ms/step - loss: 0.0048
Epoch 159/500
9/9 [==============================] - 0s 26ms/step - loss: 0.0061
Epoch 160/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0044
Epoch 161/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0043
Epoch 162/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0051: 0s - loss: 0.005
Epoch 163/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0064
Epoch 164/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0069
Epoch 165/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0064
Epoch 166/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0055
Epoch 167/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0059
Epoch 168/500
9/9 [==============================] - 0s 42ms/step - loss: 0.0045
Epoch 169/500
9/9 [==============================] - 0s 32ms/step - loss: 0.0044
Epoch 170/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0067
Epoch 171/500
9/9 [==============================] - 0s 42ms/step - loss: 0.0048
Epoch 172/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0046
Epoch 173/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0059
Epoch 174/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0084
Epoch 175/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0068
Epoch 176/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0030
Epoch 177/500
9/9 [==============================] - 0s 31ms/step - loss: 0.0054
Epoch 178/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0042
Epoch 179/500
9/9 [==============================] - 0s 43ms/step - loss: 0.0056
Epoch 180/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0092
Epoch 181/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0034
Epoch 182/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0051
Epoch 183/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0044
Epoch 184/500
9/9 [==============================] - 0s 42ms/step - loss: 0.0055
Epoch 185/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0042
Epoch 186/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0039
Epoch 187/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0037
Epoch 188/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0060
Epoch 189/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0036
Epoch 190/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0050
Epoch 191/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0034
Epoch 192/500
9/9 [==============================] - 0s 32ms/step - loss: 0.0033
Epoch 193/500
```

```
9/9 [==============================] - 0s 38ms/step - loss: 0.0041
Epoch 194/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0041
Epoch 195/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0039
Epoch 196/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0041
Epoch 197/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0029
Epoch 198/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0067
Epoch 199/500
9/9 [==============================] - 0s 30ms/step - loss: 0.0071
Epoch 200/500
9/9 [==============================] - 0s 29ms/step - loss: 0.0050
Epoch 201/500
9/9 [==============================] - 0s 25ms/step - loss: 0.0049
Epoch 202/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0042
Epoch 203/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0054
Epoch 204/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0059
Epoch 205/500
9/9 [==============================] - 0s 26ms/step - loss: 0.0037
Epoch 206/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0050
Epoch 207/500
9/9 [==============================] - 0s 26ms/step - loss: 0.0032
Epoch 208/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0044
Epoch 209/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0047
Epoch 210/500
9/9 [==============================] - 0s 42ms/step - loss: 0.0043
Epoch 211/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0056
Epoch 212/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0048
Epoch 213/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0044
Epoch 214/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0030
Epoch 215/500
9/9 [==============================] - 0s 43ms/step - loss: 0.0052
Epoch 216/500
9/9 [==============================] - 0s 44ms/step - loss: 0.0047
Epoch 217/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0043
Epoch 218/500
9/9 [==============================] - 0s 43ms/step - loss: 0.0047
Epoch 219/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0063
Epoch 220/500
9/9 [==============================] - 0s 42ms/step - loss: 0.0041
Epoch 221/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0044
Epoch 222/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0063
Epoch 223/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0049
Epoch 224/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0065
Epoch 225/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0032
Epoch 226/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0053
Epoch 227/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0039
Epoch 228/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0031
Epoch 229/500
```

```
9/9 [==============================] - 0s 35ms/step - loss: 0.0029
Epoch 230/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0035
Epoch 231/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0046
Epoch 232/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0031
Epoch 233/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0025
Epoch 234/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0044
Epoch 235/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0045
Epoch 236/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0052
Epoch 237/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0050
Epoch 238/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0034
Epoch 239/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0038
Epoch 240/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0029
Epoch 241/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0041
Epoch 242/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0036
Epoch 243/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0050
Epoch 244/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0040
Epoch 245/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0042
Epoch 246/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0037
Epoch 247/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0024
Epoch 248/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0029
Epoch 249/500
9/9 [==============================] - 0s 29ms/step - loss: 0.0033
Epoch 250/500
9/9 [==============================] - 0s 26ms/step - loss: 0.0035
Epoch 251/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0043
Epoch 252/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0035
Epoch 253/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0047
Epoch 254/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0032
Epoch 255/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0034
Epoch 256/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0054
Epoch 257/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0046
Epoch 258/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0037
Epoch 259/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0036
Epoch 260/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0059
Epoch 261/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0037
Epoch 262/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0059
Epoch 263/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0050
Epoch 264/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0061
Epoch 265/500
```

```
9/9 [==============================] - 0s 38ms/step - loss: 0.0088
Epoch 266/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0058
Epoch 267/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0084
Epoch 268/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0052
Epoch 269/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0062
Epoch 270/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0034
Epoch 271/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0039
Epoch 272/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0052
Epoch 273/500
9/9 [==============================] - 0s 31ms/step - loss: 0.0034
Epoch 274/500
9/9 [==============================] - 0s 30ms/step - loss: 0.0042
Epoch 275/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0044
Epoch 276/500
9/9 [==============================] - 0s 31ms/step - loss: 0.0028
Epoch 277/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0021
Epoch 278/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0026
Epoch 279/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0035
Epoch 280/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0049
Epoch 281/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0034
Epoch 282/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0044
Epoch 283/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0060
Epoch 284/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0040
Epoch 285/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0076
Epoch 286/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0032
Epoch 287/500
9/9 [==============================] - 0s 30ms/step - loss: 0.0039
Epoch 288/500
9/9 [==============================] - 0s 29ms/step - loss: 0.0044
Epoch 289/500
9/9 [==============================] - 0s 31ms/step - loss: 0.0059
Epoch 290/500
9/9 [==============================] - 0s 30ms/step - loss: 0.0021
Epoch 291/500
9/9 [==============================] - 0s 30ms/step - loss: 0.0036
Epoch 292/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0036
Epoch 293/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0030
Epoch 294/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0032
Epoch 295/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0043
Epoch 296/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0040
Epoch 297/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0027
Epoch 298/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0024
Epoch 299/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0038
Epoch 300/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0025
Epoch 301/500
```

```
9/9 [==============================] - 0s 36ms/step - loss: 0.0029
Epoch 302/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0028
Epoch 303/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0031
Epoch 304/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0027
Epoch 305/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0037
Epoch 306/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0033
Epoch 307/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0035
Epoch 308/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0064
Epoch 309/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0044
Epoch 310/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0050
Epoch 311/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0046
Epoch 312/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0032
Epoch 313/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0038
Epoch 314/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0032
Epoch 315/500
9/9 [==============================] - 0s 42ms/step - loss: 0.0026
Epoch 316/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0037
Epoch 317/500
9/9 [==============================] - 0s 42ms/step - loss: 0.0040
Epoch 318/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0035
Epoch 319/500
9/9 [==============================] - 0s 42ms/step - loss: 0.0039
Epoch 320/500
9/9 [==============================] - 0s 42ms/step - loss: 0.0045
Epoch 321/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0058
Epoch 322/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0043
Epoch 323/500
9/9 [==============================] - 0s 43ms/step - loss: 0.0049
Epoch 324/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0033
Epoch 325/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0038
Epoch 326/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0037
Epoch 327/500
9/9 [==============================] - 0s 45ms/step - loss: 0.0027
Epoch 328/500
9/9 [==============================] - 0s 30ms/step - loss: 0.0066
Epoch 329/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0031
Epoch 330/500
9/9 [==============================] - 0s 26ms/step - loss: 0.0027
Epoch 331/500
9/9 [==============================] - 0s 29ms/step - loss: 0.0041
Epoch 332/500
9/9 [==============================] - 0s 26ms/step - loss: 0.0029
Epoch 333/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0030
Epoch 334/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0026
Epoch 335/500
9/9 [==============================] - 0s 30ms/step - loss: 0.0020
Epoch 336/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0027
Epoch 337/500
```

```
9/9 [==============================] - 0s 35ms/step - loss: 0.0029
Epoch 338/500
9/9 [==============================] - 0s 42ms/step - loss: 0.0040
Epoch 339/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0043
Epoch 340/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0038
Epoch 341/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0029
Epoch 342/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0031
Epoch 343/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0040
Epoch 344/500
9/9 [==============================] - 0s 42ms/step - loss: 0.0032
Epoch 345/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0022
Epoch 346/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0023
Epoch 347/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0025
Epoch 348/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0031
Epoch 349/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0051
Epoch 350/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0024
Epoch 351/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0017
Epoch 352/500
9/9 [==============================] - 0s 43ms/step - loss: 0.0030
Epoch 353/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0027
Epoch 354/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0036
Epoch 355/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0031
Epoch 356/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0033
Epoch 357/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0033
Epoch 358/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0027
Epoch 359/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0041
Epoch 360/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0032
Epoch 361/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0021
Epoch 362/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0020
Epoch 363/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0030
Epoch 364/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0039
Epoch 365/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0023
Epoch 366/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0028
Epoch 367/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0018
Epoch 368/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0026
Epoch 369/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0033
Epoch 370/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0018
Epoch 371/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0031
Epoch 372/500
9/9 [==============================] - 0s 26ms/step - loss: 0.0040
Epoch 373/500
```

```
9/9 [==============================] - 0s 28ms/step - loss: 0.0047
Epoch 374/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0033
Epoch 375/500
9/9 [==============================] - 0s 30ms/step - loss: 0.0055
Epoch 376/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0032
Epoch 377/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0038
Epoch 378/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0054
Epoch 379/500
9/9 [==============================] - 0s 29ms/step - loss: 0.0030
Epoch 380/500
9/9 [==============================] - 0s 29ms/step - loss: 0.0031
Epoch 381/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0040
Epoch 382/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0035
Epoch 383/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0041
Epoch 384/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0032
Epoch 385/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0025
Epoch 386/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0035
Epoch 387/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0019
Epoch 388/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0019
Epoch 389/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0031
Epoch 390/500
9/9 [==============================] - 0s 32ms/step - loss: 0.0028
Epoch 391/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0054
Epoch 392/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0028
Epoch 393/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0029
Epoch 394/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0018
Epoch 395/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0022
Epoch 396/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0025
Epoch 397/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0031
Epoch 398/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0024: 0s - loss: 0.
Epoch 399/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0033
Epoch 400/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0019
Epoch 401/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0029
Epoch 402/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0039
Epoch 403/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0042
Epoch 404/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0037
Epoch 405/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0052
Epoch 406/500
9/9 [==============================] - 0s 44ms/step - loss: 0.0029
Epoch 407/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0021
Epoch 408/500
9/9 [==============================] - 0s 43ms/step - loss: 0.0031
Epoch 409/500
```

```
9/9 [==============================] - 0s 41ms/step - loss: 0.0022
Epoch 410/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0047
Epoch 411/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0028
Epoch 412/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0028
Epoch 413/500
9/9 [==============================] - 0s 44ms/step - loss: 0.0042
Epoch 414/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0036
Epoch 415/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0034
Epoch 416/500
9/9 [==============================] - 0s 30ms/step - loss: 0.0032
Epoch 417/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0030
Epoch 418/500
9/9 [==============================] - 0s 27ms/step - loss: 0.0038
Epoch 419/500
9/9 [==============================] - 0s 29ms/step - loss: 0.0048
Epoch 420/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0042
Epoch 421/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0025
Epoch 422/500
9/9 [==============================] - 0s 26ms/step - loss: 0.0025
Epoch 423/500
9/9 [==============================] - 0s 30ms/step - loss: 0.0039
Epoch 424/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0021
Epoch 425/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0022
Epoch 426/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0027
Epoch 427/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0029
Epoch 428/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0029
Epoch 429/500
9/9 [==============================] - 0s 43ms/step - loss: 0.0021
Epoch 430/500
9/9 [==============================] - 0s 42ms/step - loss: 0.0024
Epoch 431/500
9/9 [==============================] - 0s 42ms/step - loss: 0.0018
Epoch 432/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0029
Epoch 433/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0035
Epoch 434/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0035
Epoch 435/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0028
Epoch 436/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0027
Epoch 437/500
9/9 [==============================] - 0s 43ms/step - loss: 0.0019
Epoch 438/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0021
Epoch 439/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0034
Epoch 440/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0019
Epoch 441/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0020
Epoch 442/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0034
Epoch 443/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0014
Epoch 444/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0021
Epoch 445/500
```

```
9/9 [==============================] - 0s 34ms/step - loss: 0.0013
Epoch 446/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0032
Epoch 447/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0020
Epoch 448/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0028
Epoch 449/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0021
Epoch 450/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0022
Epoch 451/500
9/9 [==============================] - 0s 38ms/step - loss: 0.0031
Epoch 452/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0028
Epoch 453/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0025
Epoch 454/500
9/9 [==============================] - 0s 40ms/step - loss: 0.0019
Epoch 455/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0035
Epoch 456/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0018
Epoch 457/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0020
Epoch 458/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0024
Epoch 459/500
9/9 [==============================] - 0s 32ms/step - loss: 0.0036
Epoch 460/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0017
Epoch 461/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0033
Epoch 462/500
9/9 [==============================] - 0s 31ms/step - loss: 0.0027
Epoch 463/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0031
Epoch 464/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0027
Epoch 465/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0033
Epoch 466/500
9/9 [==============================] - 0s 30ms/step - loss: 0.0030
Epoch 467/500
9/9 [==============================] - 0s 28ms/step - loss: 0.0024
Epoch 468/500
9/9 [==============================] - 0s 29ms/step - loss: 0.0028
Epoch 469/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0031
Epoch 470/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0022
Epoch 471/500
9/9 [==============================] - 0s 32ms/step - loss: 0.0032
Epoch 472/500
9/9 [==============================] - 0s 32ms/step - loss: 0.0028
Epoch 473/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0030
Epoch 474/500
9/9 [==============================] - 0s 32ms/step - loss: 0.0020
Epoch 475/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0021
Epoch 476/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0022
Epoch 477/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0024
Epoch 478/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0020
Epoch 479/500
9/9 [==============================] - 0s 41ms/step - loss: 0.0026
Epoch 480/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0036
Epoch 481/500
```

```
9/9 [==============================] - 0s 38ms/step - loss: 0.0043
Epoch 482/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0052
Epoch 483/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0018
Epoch 484/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0029
Epoch 485/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0027
Epoch 486/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0029
Epoch 487/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0022
Epoch 488/500
9/9 [==============================] - 0s 31ms/step - loss: 0.0023
Epoch 489/500
9/9 [==============================] - 0s 37ms/step - loss: 0.0024
Epoch 490/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0019
Epoch 491/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0020
Epoch 492/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0032
Epoch 493/500
9/9 [==============================] - 0s 33ms/step - loss: 0.0023
Epoch 494/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0020
Epoch 495/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0020
Epoch 496/500
9/9 [==============================] - 0s 34ms/step - loss: 0.0016
Epoch 497/500
9/9 [==============================] - 0s 32ms/step - loss: 0.0020
Epoch 498/500
9/9 [==============================] - 0s 36ms/step - loss: 0.0021
Epoch 499/500
9/9 [==============================] - 0s 35ms/step - loss: 0.0025
Epoch 500/500
9/9 [==============================] - 0s 39ms/step - loss: 0.0022
```

Out[790]:

```
<tensorflow.python.keras.callbacks.History at 0x22e837fb220>
```

In [791]:

```
# testing
test1 = test.reshape(-1, 1)
```

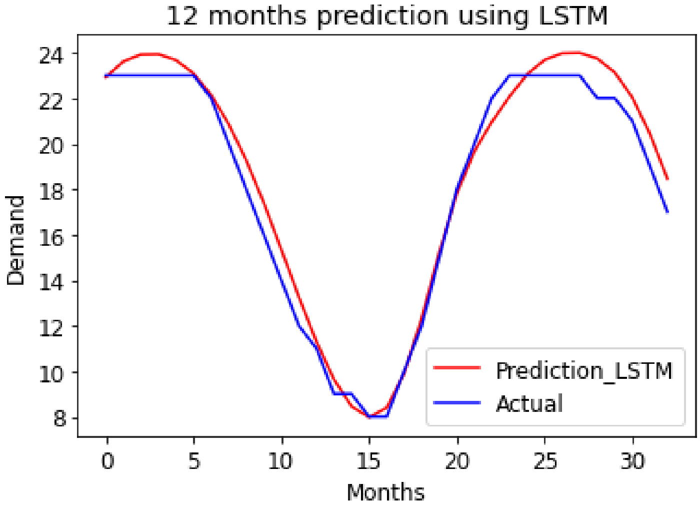
In [ ]:

```
# Getting the predicted stock price of 2017
dataset_total = np.concatenate((train1, test1), axis = 0)
inputs = dataset_total[len(dataset_total) - len(test1) - 20:]
inputs = inputs.reshape(-1,1)

inputs = sc.transform(inputs)
X_test = []
for i in range(20, 20+ len(test1)):
    X_test.append(inputs[i-20:i, 0])
X_test = np.array(X_test)
X_test = np.reshape(X_test, (X_test.shape[0], X_test.shape[1], 1))

#print(X_test.shape)
#X_test = X_test.reshape(-1, 1)

predicted_demand = regressor.predict(X_test)
predicted_demand = sc.inverse_transform(predicted_demand)
```

In [793]:

```
# Visualising the results
```

```
plt.plot(predicted_demand, color = 'red', label = 'Prediction_LSTM')
plt.plot(test1, color = 'blue', label = 'Actual')
plt.title('12 months prediction using LSTM')
plt.xlabel('Months')
plt.ylabel('Demand')
plt.legend()
plt.show()
```



In [794]:

```
MSE = np.square(np.subtract(predicted_demand, test1)).mean()
print("MSE from LSTM model is", MSE)
```

```
MSE from LSTM model is 0.7705398039637671
```

LSTM usally performs better than ARIMA but in this case we have small data and so MSE of ARIMA model was 0.61 which is better than 0.77 MSE of LSTM on test set. Using LSTM was like over killing the problem!