Time Series Project - Weather Forecast

Problem Statement:

Climate in Delhi has always been extreme with extreme heat in summers and extreme cold in winters. This aim of this project is to analyze the weather data for Delhi using various models to give a fairly accurate forecast for future weather using Python.

Data:

The source data for this project is a csv file obtained from Kaggle which contains hourly data samples from the November, 1996 to April, 2017. (01/11/1996) - (24/04/2017)

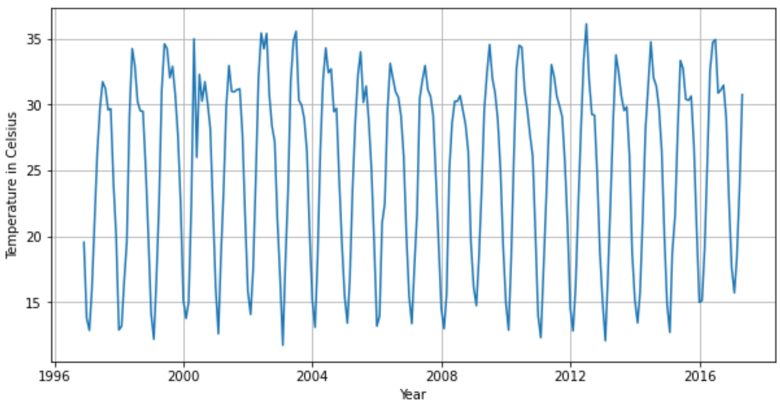
Each record contains data for 19 different weather parameters. We will be using only the temperature parameter for this project.

Process:

1. *Data Cleaning:*

- The csv file is read into Python as a Pandas data-frame. The data is parsed, only temperature is the only parameter taken, corresponding dates for temperature records are taken as the index for the data-frame.
- NA values are replaced with the respective monthly average temperature.
- Data is resampled from hourly to monthly.
- Finally, the monthly weather data looks like this:

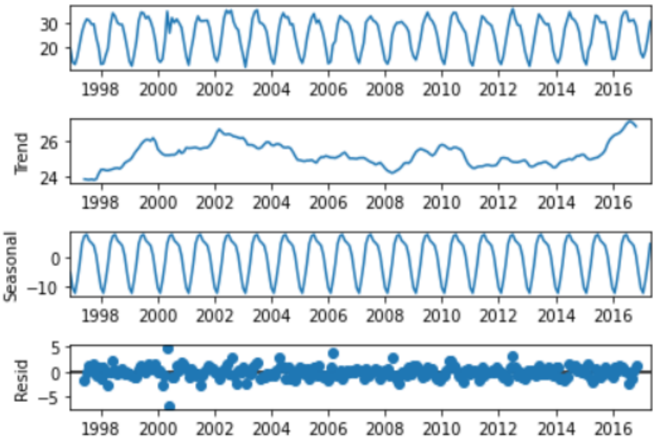
In [184]:
```
plt.figure(figsize=(10,5))
plt.plot(data_monthly.index, data_monthly['temperature'])
plt.xlabel('Year')
plt.ylabel('Temperature in Celsius')
plt.grid(True)
plt.show()
```



2. *Components of the time series:*

In [161]:

```python
from statsmodels.tsa.seasonal import seasonal_decompose



analysis = data_monthly[['temperature']].copy()


decompose_result_mult = seasonal_decompose(analysis)

trend = decompose_result_mult.trend
seasonal = decompose_result_mult.seasonal
residual = decompose_result_mult.resid

decompose_result_mult.plot();
```



3. *Stationarity:*

We check the stationarity for our clean monthly data, using the Augmented Dickey-Fuller (ADF) Test.

In [57]:

```python
check_stationarity(data_monthly)
```

```
ADF Statistic: -2.195621
p-value: 0.207838
Critical Values:
	1%: -3.459
	5%: -2.874
	10%: -2.573
Non-stationary
```

As the data is not stationary we must difference it.

In [64]:

```python
#differencing the original data_monthly time series
diff = data_monthly.diff()
diff=diff.dropna()
```

Check stationarity again.

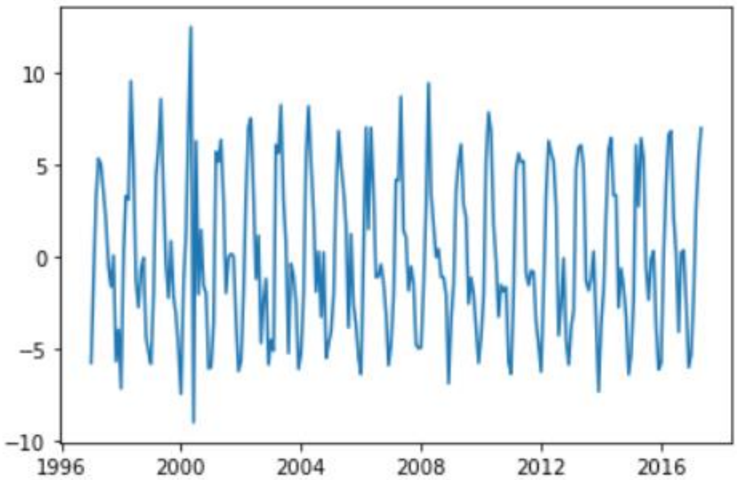
```
In [66]:  1  check_stationarity(diff)
```

```
ADF Statistic: -16.130611
p-value: 0.000000
Critical Values:
	1%: -3.459
	5%: -2.874
	10%: -2.573
Stationary
```

Our data is now stationary.

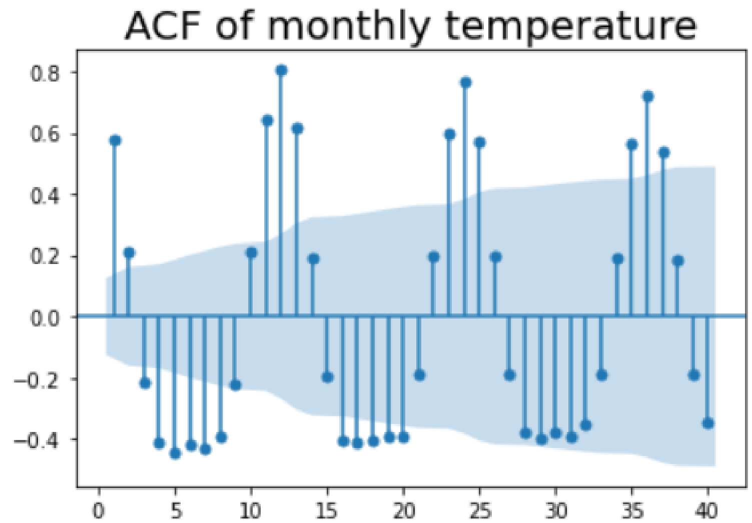
```
In [59]:  1  plt.plot(diff.index, diff['temperature'], )
```

```
Out[59]: [<matplotlib.lines.Line2D at 0x1c42cfd9f10>]
```

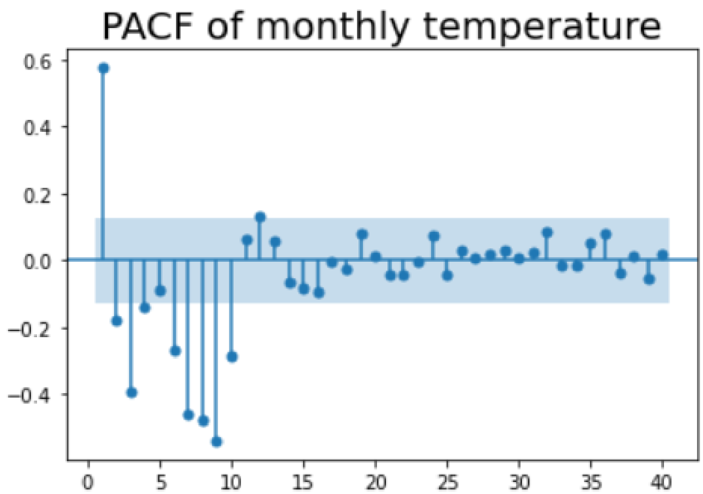


4. *ACF and PACF plots:*

In [159]:
```
plot_acf(diff, lags = 40, alpha = 0.05,zero=False)
plt.title("ACF of monthly temperature", size = 20)
plt.show()
```



In [127]:
```
plot_pacf(diff, lags = 40, alpha = 0.05 ,method='ywm', zero=False)
plt.title("PACF of monthly temperature", size = 20)
plt.show()
```

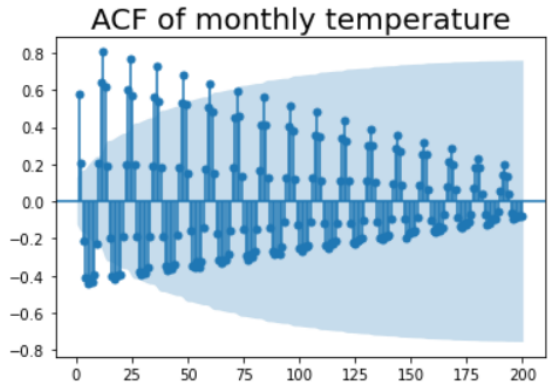


Observations and Inference drawn from the ACF, PACF plots:

- The data has seasonality.
- From the PACF plot, we can infer that value of p for an AR(p) model can be 1-9.
- MA (q) order is not clear as the ACF plot shows clustering.
- Clustering of high values followed by clustering of low values in PACF plot may mean there is an ARCH(p) component present
- Clustering of high values followed by clustering of low values in ACF plot may mean there is a GARCH(q) component present

- Seasonal order for P can be deduced by number of times the periodic clusters reach a significant value.

In [186]:
```
plot_acf(diff, lags = 200, alpha = 0.05,zero=False)
plt.title("ACF of monthly temperature", size = 20)
plt.show()
```



The clusters reach a significant value approximately 0-5 times. Order Q can be 0-5.

5. *Finding the best Model:*

We will be using the statsmodels api in python for time series models using the SARIMAX module.

First, we try to use a grid search function to brute-force and find a decent model for forecasting. Iterating over values 0,1,2 for p,d,q and P,D,Q,S, the function gives us a list of combinations with BIC as the ranking criterion.

	pdq	pdqs	bic
143	(0, 1, 2)	(0, 2, 2, 12)	759.789695
467	(1, 2, 2)	(0, 2, 2, 12)	762.299195
224	(0, 2, 2)	(0, 2, 2, 12)	763.475984
359	(1, 1, 1)	(0, 2, 2, 12)	763.914267
386	(1, 1, 2)	(0, 2, 2, 12)	763.916471

As the BIC is almost similar, we take the combination with highest AIC.

In [188]:
```
SARIMAXmodel = SARIMAX(train, order = (1, 1, 1), seasonal_order=(0,2,2,12))
results = SARIMAXmodel.fit()
SARIMAXmodel = SARIMAXmodel.fit()
SARIMAXmodel.aic
```

Out[188]: 742.1090275605816

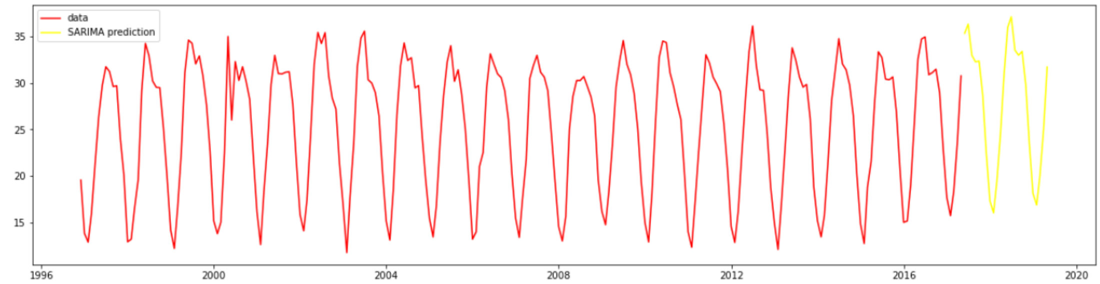
Plotting the forecast for this SARIMA model.

In [50]:

```
y_pred = SARIMAXmodel.get_forecast(len(prediction_index))
y_pred_df = y_pred.conf_int(alpha = 0.05)
y_pred_df["Predictions"] = SARIMAXmodel.predict(start = y_pred_df.index[0], end = y_pred_df.index[-1])
y_pred_df.index = prediction_index
y_pred_out = y_pred_df["Predictions"]
```

In [51]:

```
plt.figure(figsize=(20,5))

plt.plot(data_monthly, color='red', label='data')
plt.plot(y_pred_out, color='yellow', label='SARIMA prediction')

plt.legend()
plt.show()
```



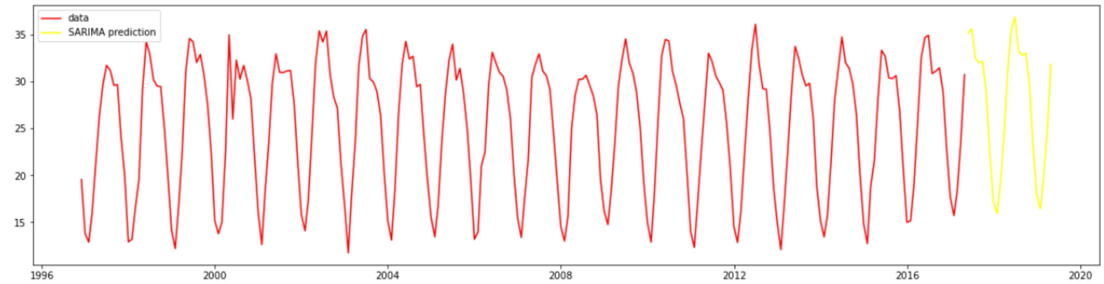
We remove the seasonality from the data and using auto arima, we find the values of p,q to be 1,1.

```
ARIMA(1,0,1) with non-zero mean
```

Using this p,q value and as we differenced the data once to make it stationary , we get d=1.
Checking AIC for seasonal orders using our inferences from the ACF and PACF plots, we get an even better model with seasonal orders (3,2,4,12).

In [193]:

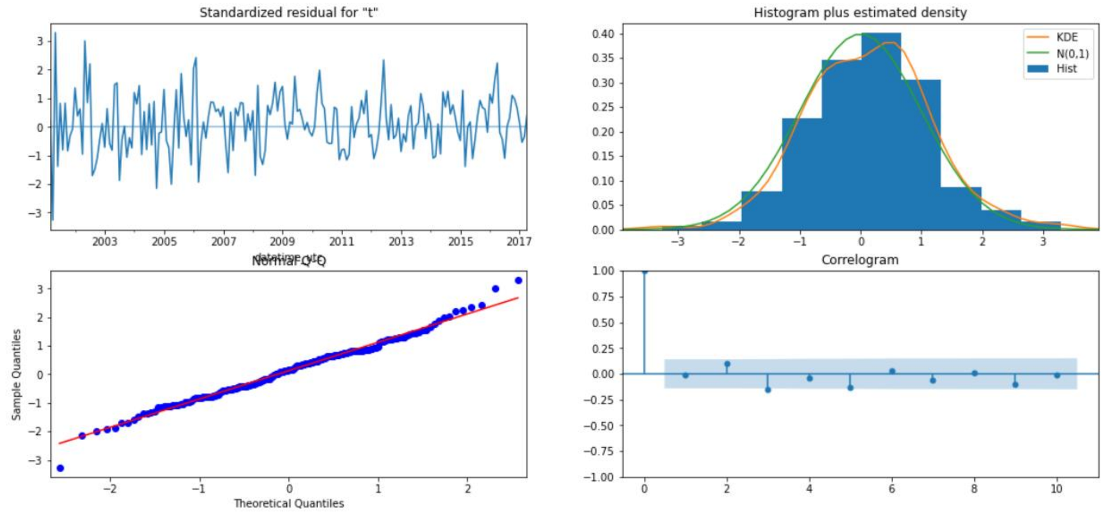
```
plt.figure(figsize=(20,5))

plt.plot(data_monthly, color='red', label='data')
plt.plot(y_pred_out, color='yellow', label='SARIMA prediction')

plt.legend()
plt.show()
```



This is a similar looking plot but it is slightly improved.

Plotting the diagnostic graphs.

In [52]:
```
results.plot_diagnostics(figsize=(18, 8))
plt.show()
```



The Q-Q plot of residuals, all observations fall close throughout the line, indicating normally distributed errors.

The histogram and density plot have a normal distribution shape and a mean of zero.

The residuals line plot fluctuates around zero with a constant variance, suggesting the residuals are white noise.

The ACF lags are within the threshold limits, indicating no autocorrelation between residual errors.