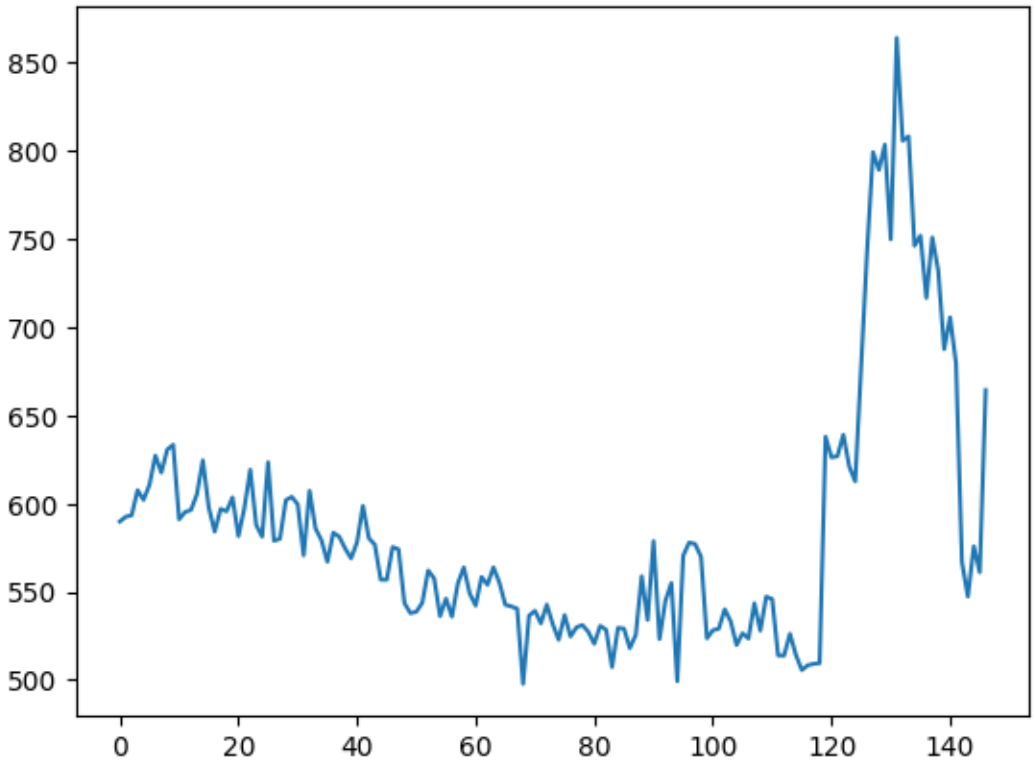
```
/opt/homebrew/lib/python3.11/site-
packages/statsmodels/tsa/holtwinters/model.py:915: ConvergenceWarning:
Optimization failed to converge. Check mle_retvals.
  warnings.warn(
/opt/homebrew/lib/python3.11/site-
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
/opt/homebrew/lib/python3.11/site-
packages/statsmodels/tsa/holtwinters/model.py:915: ConvergenceWarning:
Optimization failed to converge. Check mle_retvals.
  warnings.warn(
/opt/homebrew/lib/python3.11/site-
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
/opt/homebrew/lib/python3.11/site-
packages/statsmodels/tsa/holtwinters/model.py:915: ConvergenceWarning:
Optimization failed to converge. Check mle_retvals.
  warnings.warn(
/opt/homebrew/lib/python3.11/site-
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
/opt/homebrew/lib/python3.11/site-
packages/statsmodels/tsa/holtwinters/model.py:915: ConvergenceWarning:
Optimization failed to converge. Check mle_retvals.
  warnings.warn(
/opt/homebrew/lib/python3.11/site-
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
/opt/homebrew/lib/python3.11/site-
packages/statsmodels/tsa/holtwinters/model.py:915: ConvergenceWarning:
Optimization failed to converge. Check mle_retvals.
  warnings.warn(
```

[123]:

```
plt.plot(rmse)
plt.show()
print('Optimal Seasonal Period: ',np.argmin(rmse))
```

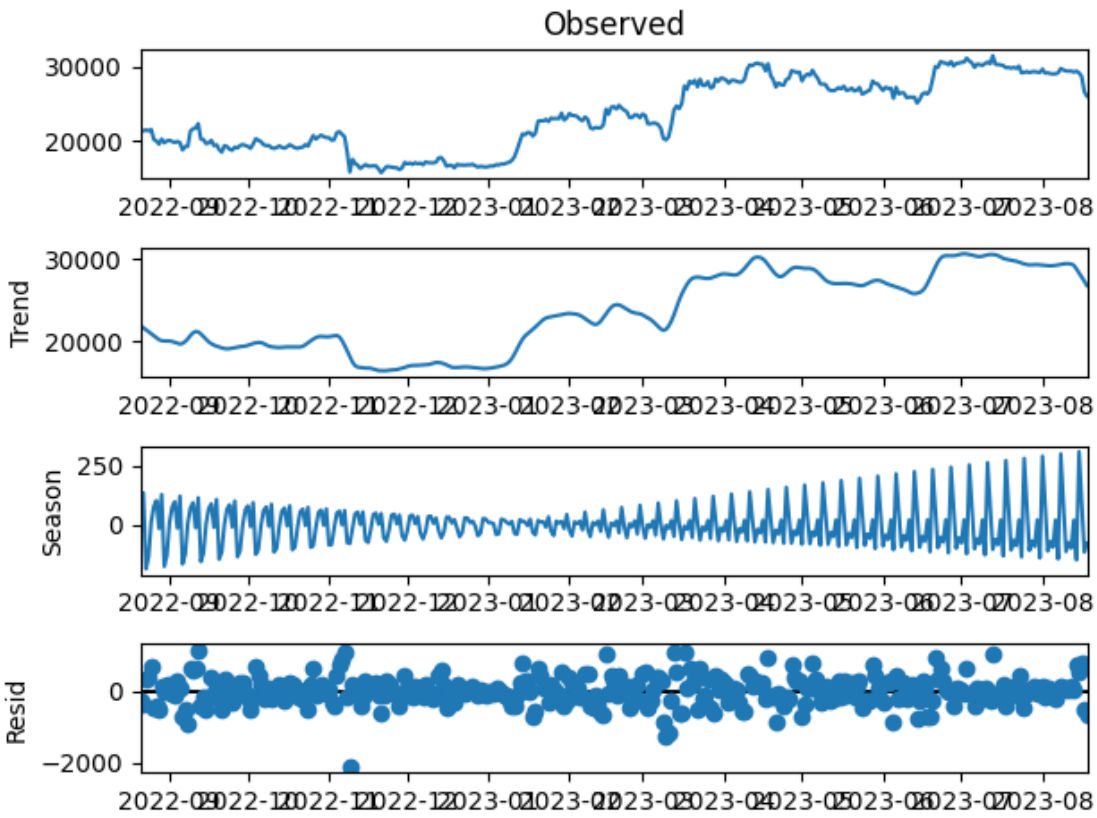


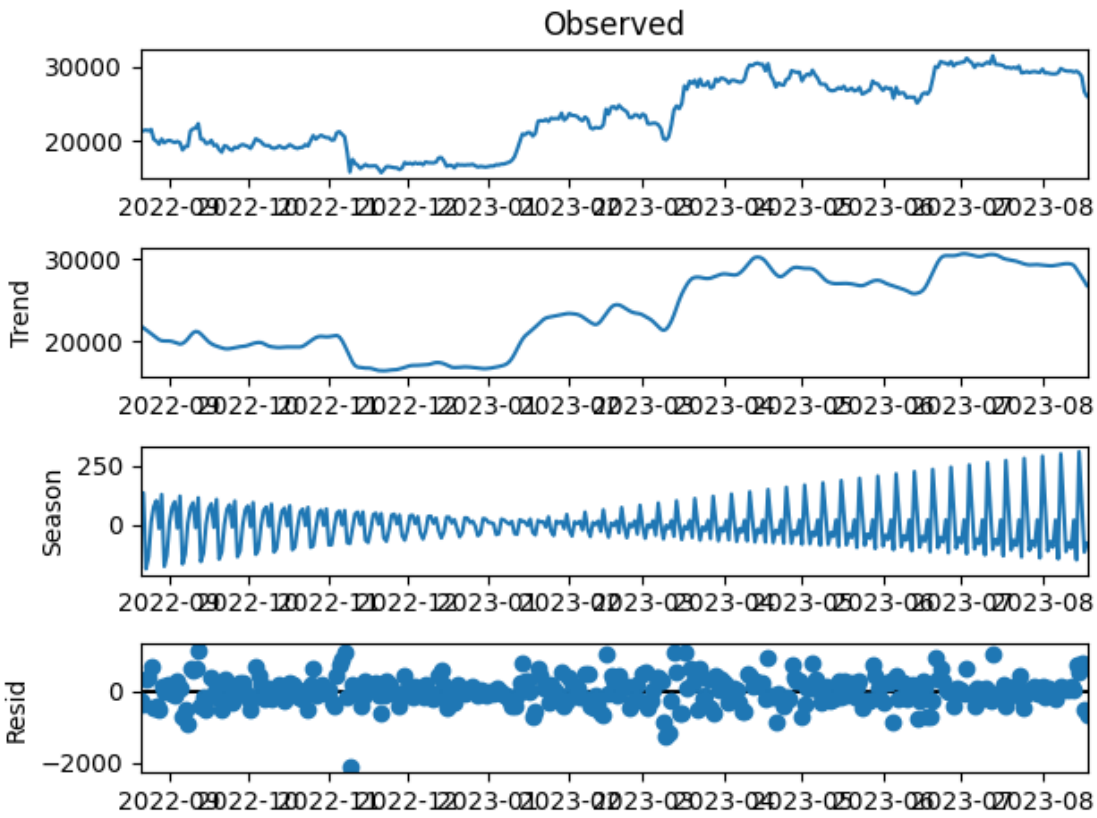

```
Optimal Seasonal Period:  68
```

We will run the STL model and plot the model to see the seasonality in the data, which we can see with a gradual decrease and increase in the variance there can be similarity in patterns each 116 days!

[124]:
```
stl_model = STL(bitcoin_data,seasonal=115).fit()

stl_model.plot()
```

[124]:


Observed
30000
20000
Trend
30000
20000
Season
250
0
Resid
0
−2000



2 Exponential Smoothing and Holt's Method:

after we understood that there is no seasonality and we have a trend we perform an exponential smoothing model with a damped trend and let the model decide the parameter values we forecast for the next 31 days which resulted in a downward trend which if we visually compare the trend with the past we can see a similar downward pattern in May 2023 up to July.

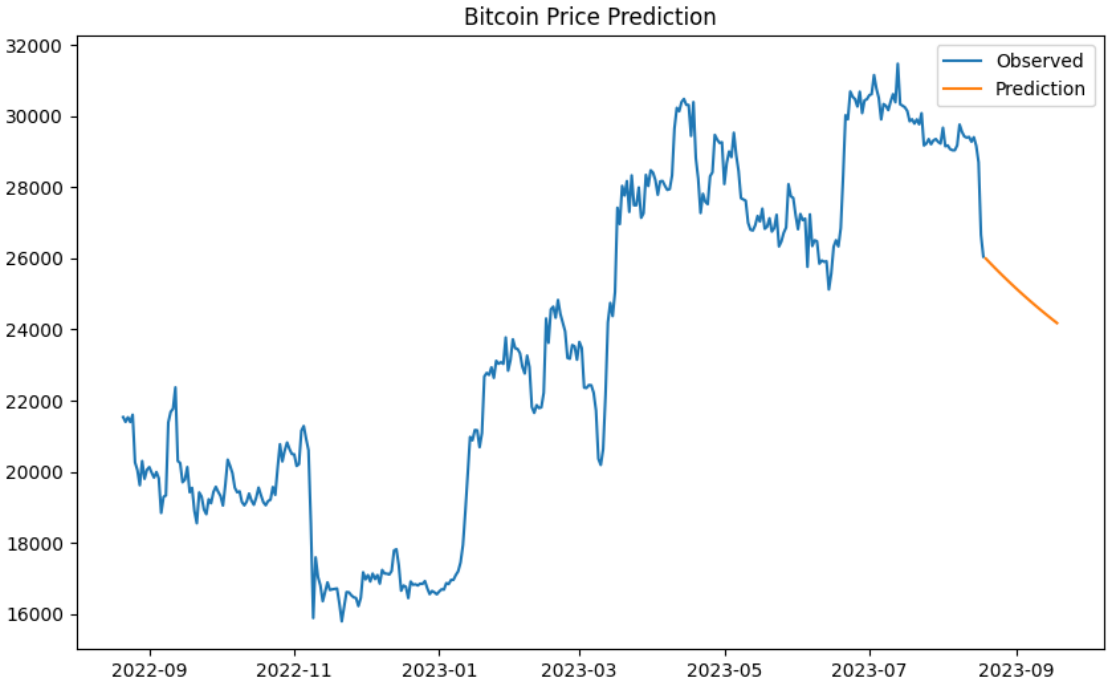
[125]:
```
modelexp=ExponentialSmoothing(bitcoin_data,
                              trend='add',
                              damped_trend=True,
                              initialization_method='estimated').fit()

modelexp.summary()

expforecast=modelexp.forecast(steps=31)
print(expforecast)
plt.figure(figsize=(10,6))
plt.plot(bitcoin_data, label='Observed')
plt.plot(expforecast, label='Prediction')
plt.title('Bitcoin Price Prediction')
```

```
plt.legend()
```

```
2023-08-19    25996.669530
2023-08-20    25926.890601
2023-08-21    25857.809461
2023-08-22    25789.419133
2023-08-23    25721.712708
2023-08-24    25654.683347
2023-08-25    25588.324280
2023-08-26    25522.628803
2023-08-27    25457.590282
2023-08-28    25393.202145
2023-08-29    25329.457890
2023-08-30    25266.351077
2023-08-31    25203.875333
2023-09-01    25142.024346
2023-09-02    25080.791868
2023-09-03    25020.171716
2023-09-04    24960.157765
2023-09-05    24900.743954
2023-09-06    24841.924280
2023-09-07    24783.692804
2023-09-08    24726.043642
2023-09-09    24668.970972
2023-09-10    24612.469029
2023-09-11    24556.532104
2023-09-12    24501.154550
2023-09-13    24446.330770
2023-09-14    24392.055229
2023-09-15    24338.322443
2023-09-16    24285.126985
2023-09-17    24232.463481
2023-09-18    24180.326613
Freq: D, dtype: float64
```

```
/opt/homebrew/lib/python3.11/site-
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
/opt/homebrew/lib/python3.11/site-
packages/statsmodels/tsa/holtwinters/model.py:915: ConvergenceWarning:
Optimization failed to converge. Check mle_retvals.
  warnings.warn(
```

[125]: <matplotlib.legend.Legend at 0x28c322550>



the results for this model are very promising which we can see next when we calculated the RMSE for this model it is the best performance among all the models we already tried. and we also can see a significant drop in values of AIC, BIC and AICc.

[126]:
```
print(modelexp.summary())
```

```
                       ExponentialSmoothing Model Results
==============================================================================
Dep. Variable:                      Price   No. Observations:                  363
Model:             ExponentialSmoothing   SSE                      123887343.190
Optimized:                         True   AIC                           4634.794
Trend:                         Additive   BIC                           4654.266
Seasonal:                          None   AICC                          4635.110
Seasonal Periods:                  None   Date:                 Mon, 21 Aug 2023
Box-Cox:                          False   Time:                         14:25:16
Box-Cox Coeff.:                    None
==============================================================================
                         coeff                 code              optimized
------------------------------------------------------------------------------
smoothing_level            0.9714286                alpha                 True
smoothing_trend            0.0231293                 beta                 True
initial_level              22023.685                  l.0                 True
initial_trend             -229.79584                  b.0                 True
damping_trend              0.9900000                  phi                 True
------------------------------------------------------------------------------
```

[127]:
```
print('RMSE for this model is: ', np.sqrt((modelexp.resid**2).mean()))
```

```
RMSE for this model is:  584.1981229035348
```

2.1 Exponential Smoothing

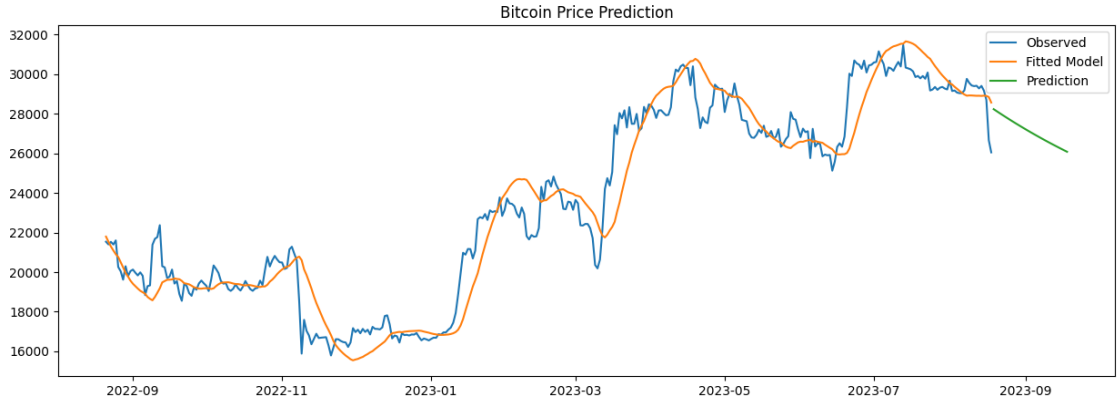
we do this model once more using "heuristic" initialization method and smoothing level and smoothing trend values of 0.1 to get a smoother model fit to see the trend. As a remark we expect to see a possible higher RMSE because it is not the optimal model.

[128]:
```
modelexp=ExponentialSmoothing(bitcoin_data,
                              trend='add',
                              damped_trend=True,
                              initialization_method='heuristic').
 ↪fit(smoothing_level=0.1,
                                                                  ␣
 ↪smoothing_trend=0.1)

modelexp.summary()

expforecast=modelexp.forecast(steps=31)
print(expforecast)

plt.figure(figsize=(15,5))
plt.plot(bitcoin_data, label='Observed')
plt.plot(modelexp.fittedvalues, label='Fitted Model')
plt.plot(expforecast, label='Prediction')
plt.title('Bitcoin Price Prediction')
plt.legend()

print('RMSE for this model is: ', np.sqrt((modelexp.resid**2).mean()))
```

```
/opt/homebrew/lib/python3.11/site-
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
/opt/homebrew/lib/python3.11/site-
packages/statsmodels/tsa/holtwinters/model.py:915: ConvergenceWarning:
Optimization failed to converge. Check mle_retvals.
  warnings.warn(
```

```
2023-08-19    28233.198357
2023-08-20    28150.505068
2023-08-21    28068.638712
2023-08-22    27987.591019
2023-08-23    27907.353804
2023-08-24    27827.918961
2023-08-25    27749.278466
2023-08-26    27671.424376
```

```
2023-08-27    27594.348827
2023-08-28    27518.044033
2023-08-29    27442.502288
2023-08-30    27367.715960
2023-08-31    27293.677495
2023-09-01    27220.379415
2023-09-02    27147.814315
2023-09-03    27075.974867
2023-09-04    27004.853813
2023-09-05    26934.443969
2023-09-06    26864.738224
2023-09-07    26795.729537
2023-09-08    26727.410936
2023-09-09    26659.775522
2023-09-10    26592.816461
2023-09-11    26526.526991
2023-09-12    26460.900416
2023-09-13    26395.930107
2023-09-14    26331.609500
2023-09-15    26267.932100
2023-09-16    26204.891474
2023-09-17    26142.481254
2023-09-18    26080.695136
Freq: D, dtype: float64
RMSE for this model is:  1312.5987110865674
```



2.2 Holt's Method

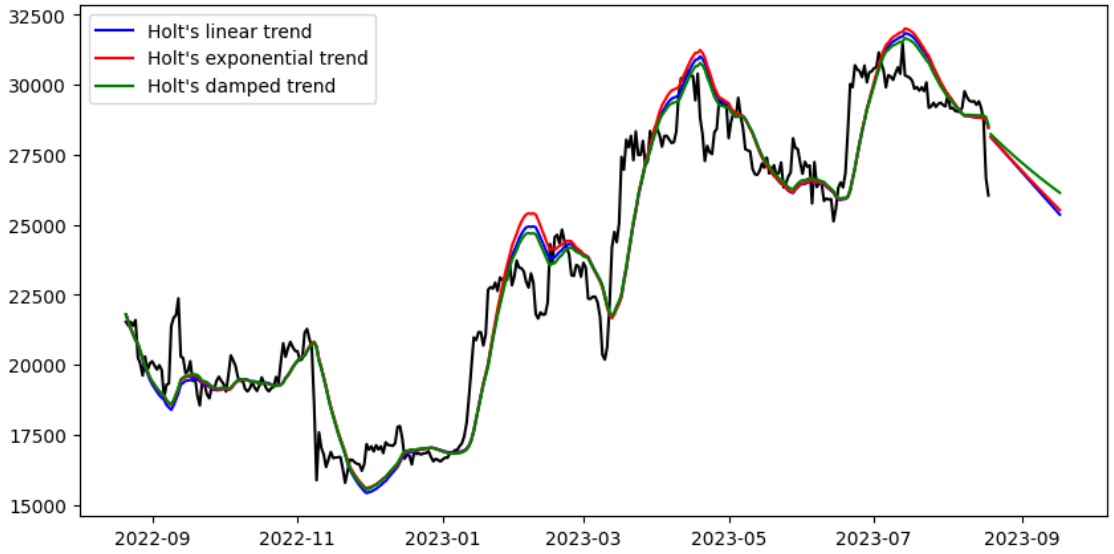
we performed 3 different methods the first one is Holt's linear method, the next one is the exponential method and last but not least Holt's damped trend model. we build all the models with smoothing level and smoothing trend values of 0.1 to compare them visually and using RMSE we can see that the lowest RMSE belongs to the damped trend.

[129]:
```
from statsmodels.tsa.api import ExponentialSmoothing,SimpleExpSmoothing,Holt
holt_1=Holt(bitcoin_data,
                    initialization_method='heuristic').fit(smoothing_level=0.1,
                                                           smoothing_trend=0.1,
optimized=False)
H_forecast1= holt_1.forecast(30).rename("Holt's linear trend")
holt_2=Holt(bitcoin_data, exponential=True,
                    initialization_method='heuristic').fit(smoothing_level=0.1,
                                                           smoothing_trend=0.1,
optimized=False)
H_forecast2= holt_2.forecast(30).rename("Holt's exponential trend")
holt_3=Holt(bitcoin_data, damped_trend=True,
                    initialization_method='heuristic').fit(smoothing_level=0.1,
                                                           smoothing_trend=0.1)
H_forecast3= holt_3.forecast(30).rename("Holt's damped trend")
plt.figure(figsize=(10,5))
plt.plot(bitcoin_data, color='black')
plt.plot(holt_1.fittedvalues, color='blue')
(line1,) = plt.plot(H_forecast1,color='blue')
plt.plot(holt_2.fittedvalues, color='red')
(line2,) = plt.plot(H_forecast2,color='red')
plt.plot(holt_3.fittedvalues, color='green')
(line3,) = plt.plot(H_forecast3,color='green')
plt.legend([line1,line2,line3],[H_forecast1.name,H_forecast2.name,H_forecast3.
 ↪name])
print('RMSE linear model',np.sqrt(((holt_1.resid)**2). mean()))
print('RMSE exponential model',np.sqrt(((holt_2. resid)**2).mean()))
print('RMSE damped model',np.sqrt(((holt_3.resid)**2). mean()))
```

```
RMSE linear model 1366.5381636030627
RMSE exponential model 1403.5148100766453
RMSE damped model 1312.5987110865674
```

```
/opt/homebrew/lib/python3.11/site-
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
/opt/homebrew/lib/python3.11/site-
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
/opt/homebrew/lib/python3.11/site-
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
/opt/homebrew/lib/python3.11/site-
packages/statsmodels/tsa/holtwinters/model.py:915: ConvergenceWarning:
Optimization failed to converge. Check mle_retvals.
```

```
warnings.warn(
```



2.3 ETS Model

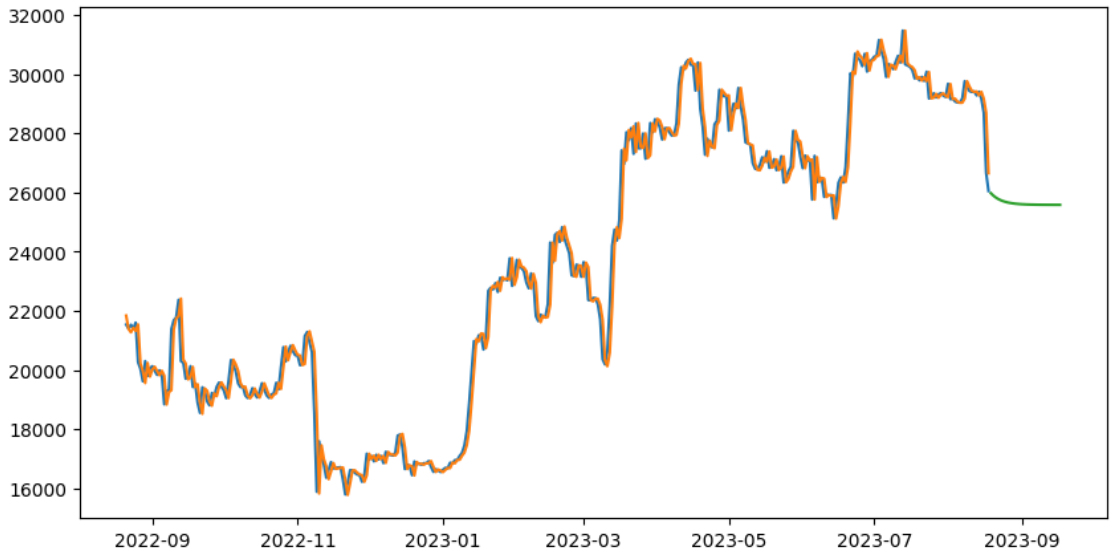
Lastly, we will try an ETS model with an additive damped trend and let the algorithm estimate the parameters

[130]:
```
from statsmodels.tsa.exponential_smoothing.ets import ETSModel

ets = ETSModel(bitcoin_data['Price'],
               trend='add',
               damped_trend=True,
               initialization_method='estimated').fit()

ets_pred=ets.forecast(30)
plt.figure(figsize=(10,5))
plt.plot(bitcoin_data)
plt.plot(ets.fittedvalues)
plt.plot(ets_pred)
```

```
RUNNING THE L-BFGS-B CODE

           * * *

Machine precision = 2.220D-16
 N =            5     M =           10

At X0         1 variables are exactly at the bounds
```

```
At iterate    0    f=  8.56853D+00    |proj g|=  8.99900D-01

At iterate    1    f=  7.79002D+00    |proj g|=  1.23181D-01

At iterate    2    f=  7.78498D+00    |proj g|=  7.02062D-02

At iterate    3    f=  7.78249D+00    |proj g|=  1.31566D-02

At iterate    4    f=  7.78238D+00    |proj g|=  9.78391D-03

At iterate    5    f=  7.78212D+00    |proj g|=  4.36469D-03

At iterate    6    f=  7.78211D+00    |proj g|=  5.04841D-04

At iterate    7    f=  7.78211D+00    |proj g|=  2.84217D-06

           * * *

Tit   = total number of iterations
Tnf   = total number of function evaluations
Tnint = total number of segments explored during Cauchy searches
Skip  = number of BFGS updates skipped
Nact  = number of active bounds at final generalized Cauchy point
Projg = norm of the final projected gradient
F     = final function value

           * * *

   N    Tit     Tnf  Tnint  Skip  Nact     Projg        F
    5      7      8      9     0     1   2.842D-06   7.782D+00
  F =   7.7821059691958290

CONVERGENCE: NORM_OF_PROJECTED_GRADIENT_<=_PGTOL

/opt/homebrew/lib/python3.11/site-
packages/statsmodels/tsa/base/tsa_model.py:471: ValueWarning: No frequency
information was provided, so inferred frequency D will be used.
  self._init_dates(dates, freq)
```

[130]: [<matplotlib.lines.Line2D at 0x28c528a50>]



ETS model performed the best among all the models considering RMSE which we can see in the last code chunk, and from the chart we saw earlier predicts the price to continue the downward trend and settles around the value of 25500 in the next 30 days.

[131]:
```
print(ets_pred)
```

```
2023-08-19    25974.955396
2023-08-20    25896.304966
2023-08-21    25833.384622
2023-08-22    25783.048346
2023-08-23    25742.779326
2023-08-24    25710.564110
2023-08-25    25684.791937
2023-08-26    25664.174198
2023-08-27    25647.680007
2023-08-28    25634.484655
2023-08-29    25623.928373
2023-08-30    25615.483347
2023-08-31    25608.727327
2023-09-01    25603.322510
2023-09-02    25598.998657
2023-09-03    25595.539575
2023-09-04    25592.772309
2023-09-05    25590.558496
2023-09-06    25588.787446
2023-09-07    25587.370605
2023-09-08    25586.237133
2023-09-09    25585.330355
2023-09-10    25584.604933
```

```
2023-09-11    25584.024596
2023-09-12    25583.560325
2023-09-13    25583.188909
2023-09-14    25582.891776
2023-09-15    25582.654070
2023-09-16    25582.463905
2023-09-17    25582.311773
Freq: D, Name: simulation, dtype: float64
```

[132]:
```
print(ets.summary())
```

```
                                ETS Results
==============================================================================
Dep. Variable:                   Price   No. Observations:                  363
Model:                     ETS(AAdN)   Log Likelihood               -2824.904
Date:                Mon, 21 Aug 2023   AIC                           5661.809
Time:                        14:25:17   BIC                           5685.175
Sample:                    08-21-2022   HQIC                          5671.097
                         - 08-18-2023   Scale                       336493.756
Covariance Type:               approx
==============================================================================
                     coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
---
smoothing_level     0.9605        nan        nan        nan         nan
nan
smoothing_trend     0.0467        nan        nan        nan         nan
nan
damping_trend       0.8000        nan        nan        nan         nan
nan
initial_level    2.202e+04    646.794     34.051      0.000    2.08e+04
2.33e+04
initial_trend    -232.1170    461.987     -0.502      0.615   -1137.595
673.361
==============================================================================
===
Ljung-Box (Q):                        1.28   Jarque-Bera (JB):
234.50
Prob(Q):                              0.53   Prob(JB):
0.00
Heteroskedasticity (H):               0.91   Skew:
0.10
Prob(H) (two-sided):                  0.59   Kurtosis:
6.93
==============================================================================
===
```

```
Warnings:
[1] Covariance matrix calculated using numerical (complex-step) differentiation.
```

[133]:
```
print("RMSE for ETS Model: ", np.sqrt((ets.resid**2).mean()))
```

```
RMSE for ETS Model:  580.0808184515867
```

3 Conclusion

We saw that the ETS model performed best considering RMSE and predicted a more constant trend in the next 30 days but the first exponential smoothing method that we did we received a slightly higher RMSE from the model but the values of AIC, BIC and AICc were very lower than ETS model. then we choose the best model to be an exponential smoothing or Holt's damped trend model.