```
import pandas as pd
from matplotlib import pyplot as plt
import seaborn as sns
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose
import statsmodels.graphics.tsaplots as tsa_plots
from statsmodels.tsa.arima_model import ARIMA
from matplotlib import pyplot
from sklearn.metrics import mean_squared_error
import numpy as np
import statsmodels.formula.api as smf
from datetime import datetime,time
```

Data Collection

```
df=pd.read_excel('Airlines+Data.xlsx')
```

```
df1=df.copy()
```

```
df1.head()
```

	Month	Passengers
0	1995-01-01	112
1	1995-02-01	118
2	1995-03-01	132
3	1995-04-01	129
4	1995-05-01	121

```
df1.isnull().sum()
```

```
Month         0
Passengers    0
dtype: int64
```

```
df1.dtypes
```

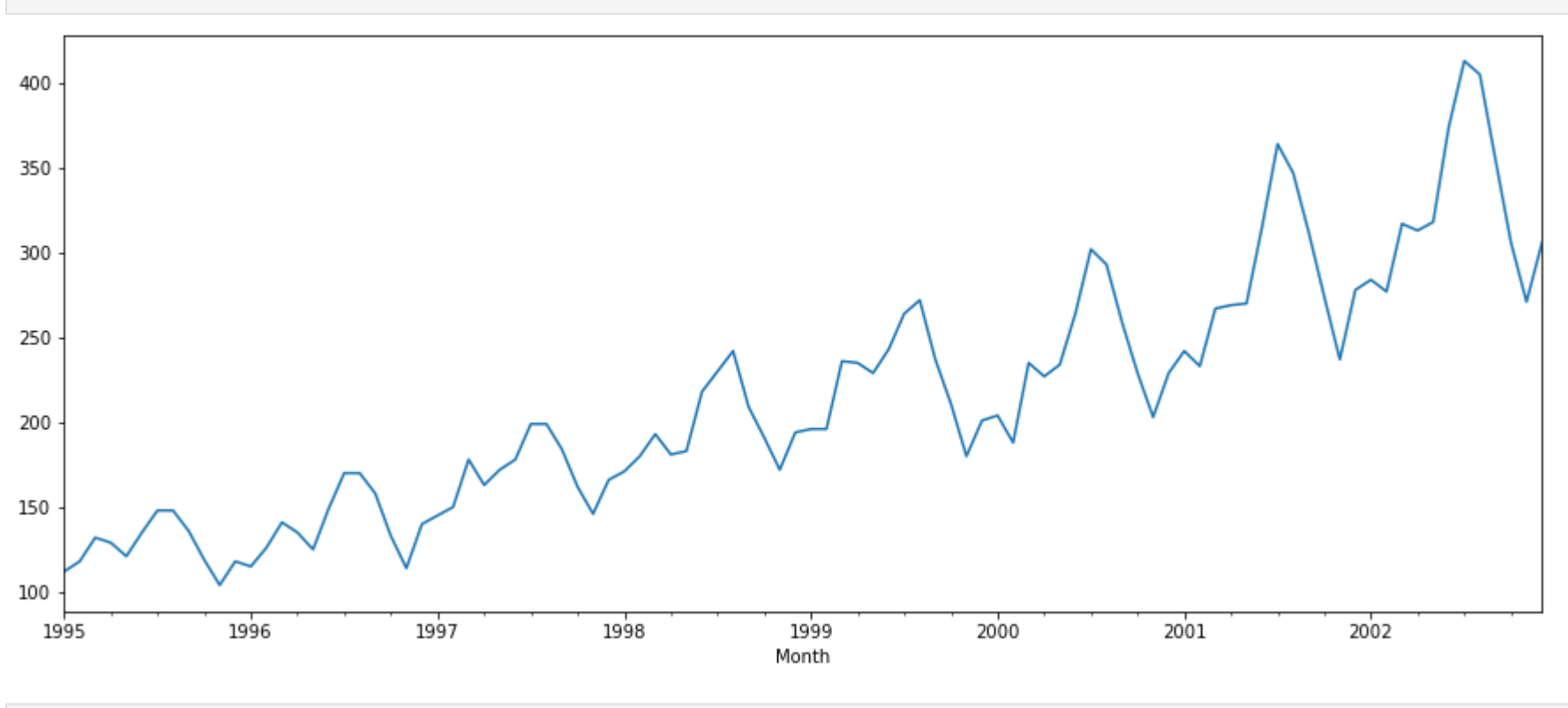
```
Month         datetime64[ns]
Passengers             int64
dtype: object
```

```
df1.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Passengers	96.0	213.708333	71.918216	104.0	156.0	200.0	264.75	413.0

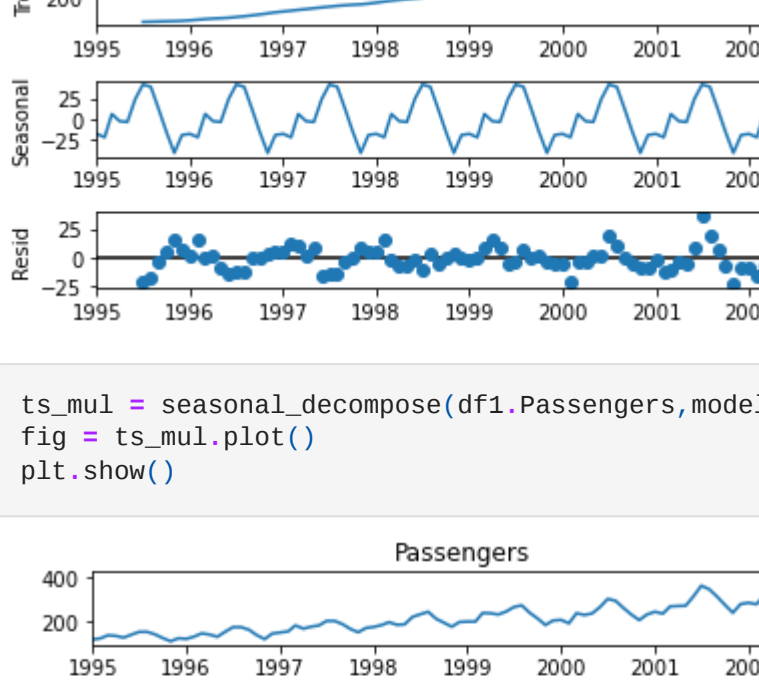
```
df1 = df1.set_index('Month')
```

```
df1['Passengers'].plot(figsize=(15, 6))
plt.show()
```

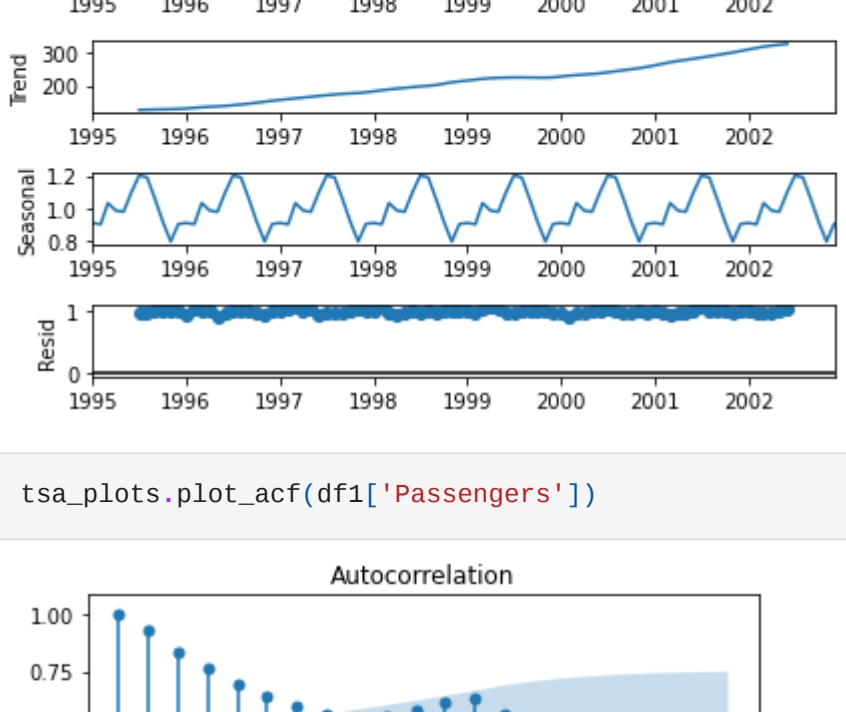


```
for i in range(2,10,2):
    df1['Passengers'].rolling(i).mean().plot(label=str(i))
plt.legend(loc=3)
```

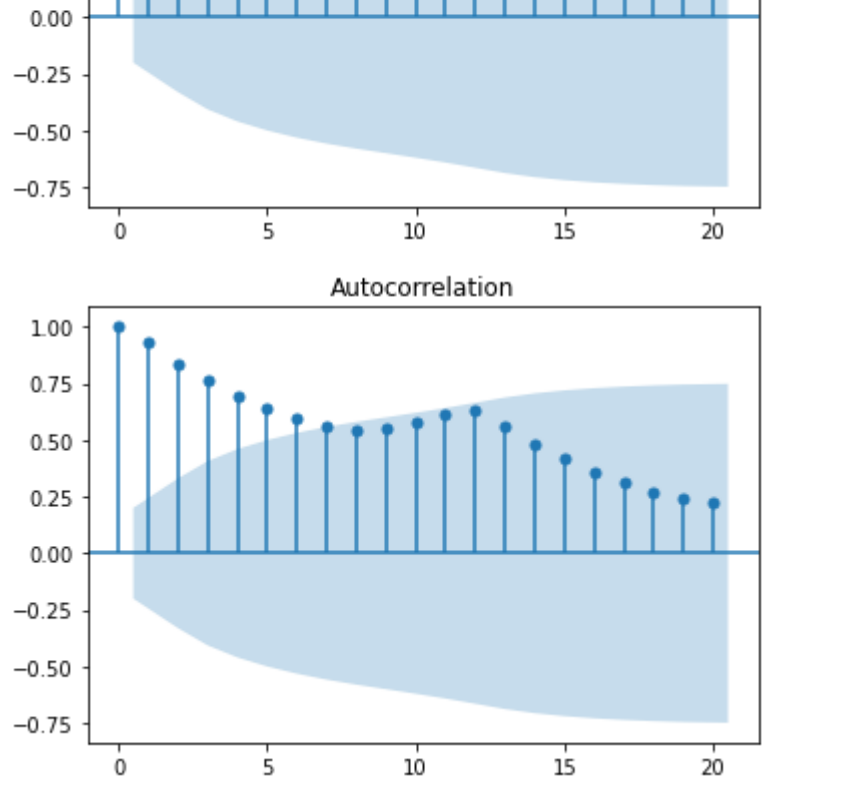
<matplotlib.legend.Legend at 0x20498244760>



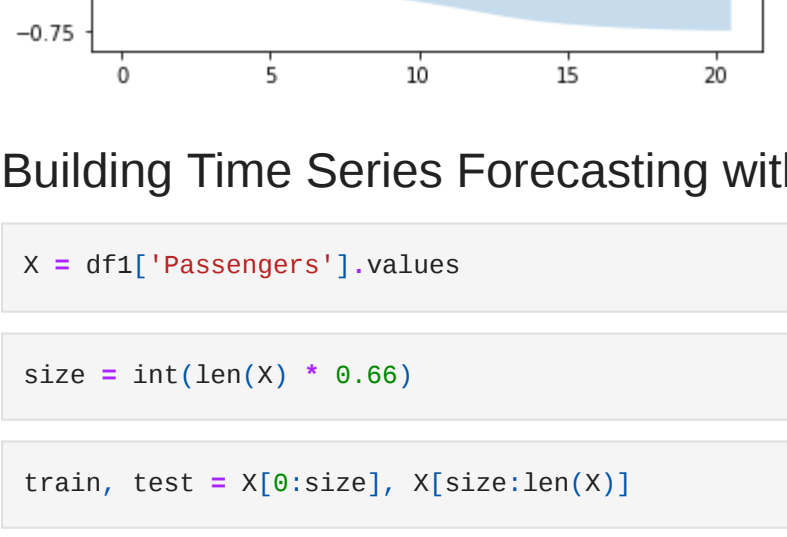
```
ts_add = seasonal_decompose(df1['Passengers'],model="additive")
fig = ts_add.plot()
plt.show()
```

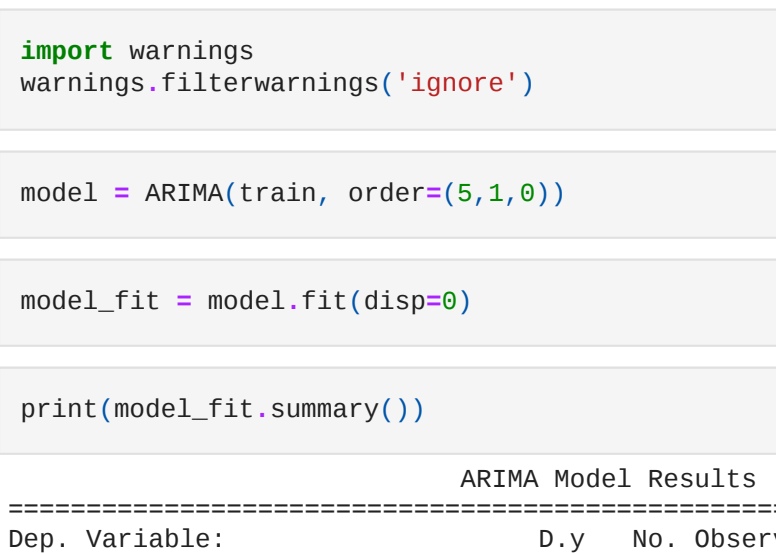



```
ts_mul = seasonal_decompose(df1.Passengers,model="multiplicative")
fig = ts_mul.plot()
plt.show()
```




```
tsa_plots.plot_acf(df1['Passengers'])
```





Building Time Series Forecasting with ARIMA

```
X = df1['Passengers'].values
```

```
size = int(len(X) * 0.66)
```

```
train, test = X[0:size], X[size:len(X)]
```

```
import warnings
warnings.filterwarnings('ignore')
```

```
model = ARIMA(train, order=(5,1,0))
```

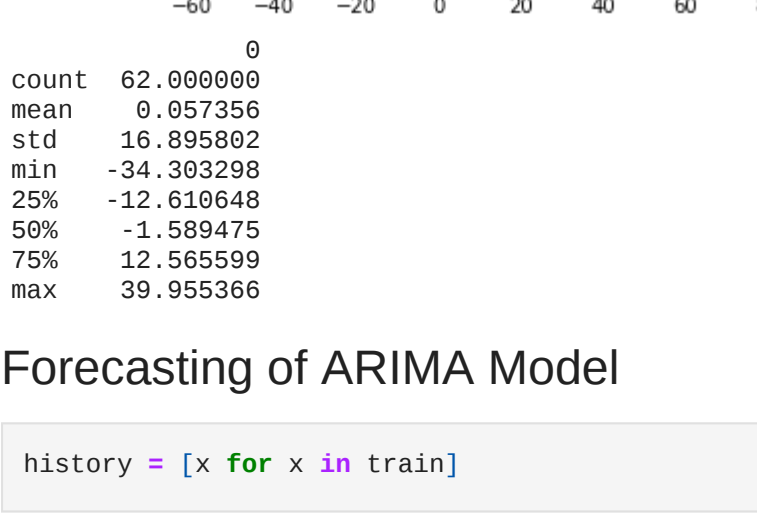
```
model_fit = model.fit(disp=0)
```

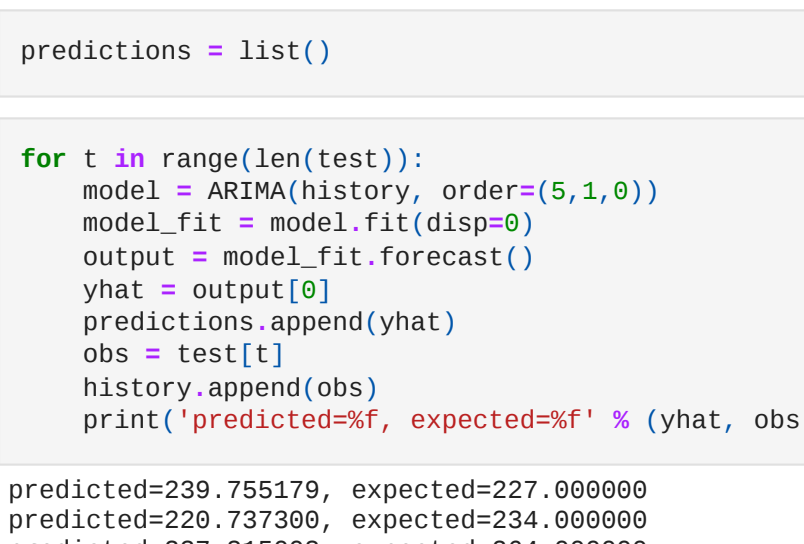
```
print(model_fit.summary())
```

```
                             ARIMA Model Results
==============================================================================
Dep. Variable:                    D.y   No. Observations:                   62
Model:                 ARIMA(5, 1, 0)   Log Likelihood                -262.909
Method:                       css-mle   S.D. of innovations             16.748
Date:                Sat, 25 Dec 2021   AIC                            539.817
Time:                        16:06:40   BIC                            554.707
Sample:                             1   HQIC                           545.663

==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
const          1.7497      1.477      1.185      0.236      -1.145       4.644
ar.L1.D.y      0.0905      0.134      0.677      0.498      -0.171       0.352
ar.L2.D.y     -0.2096      0.135     -1.549      0.121      -0.475       0.056
ar.L3.D.y     -0.0829      0.133     -0.623      0.533      -0.344       0.178
ar.L4.D.y     -0.2598      0.133     -1.953      0.051      -0.521       0.001
ar.L5.D.y     -0.0090      0.139     -0.065      0.948      -0.282       0.264
                                    Roots
=============================================================================
                  Real          Imaginary           Modulus         Frequency
-----------------------------------------------------------------------------
AR.1            0.8182           -1.0121j            1.3015           -0.1418
AR.2            0.8182           +1.0121j            1.3015            0.1418
AR.3           -0.9648           -1.1683j            1.5152           -0.3599
AR.4           -0.9648           +1.1683j            1.5152            0.3599
AR.5          -28.5048           -0.0000j           28.5048           -0.5000
-----------------------------------------------------------------------------
```

```
residuals = pd.DataFrame(model_fit.resid)
residuals.plot()
pyplot.show()
residuals.plot(kind='kde')
pyplot.show()
print(residuals.describe())
```






```
               0
count  62.000000
mean    0.057356
std    16.895802
min   -34.303298
25%   -12.610648
50%    -1.589475
75%    12.565599
max    39.955366
```

Forecasting of ARIMA Model

```
history = [x for x in train]
```

```
predictions = list()
```

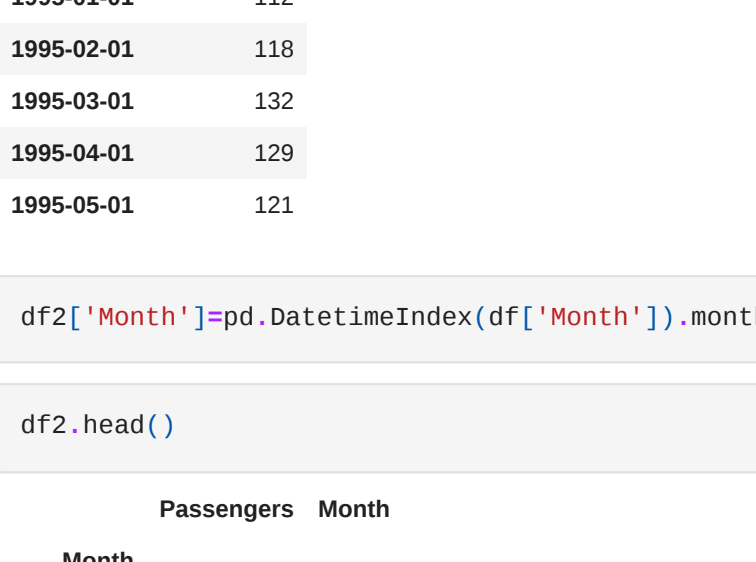
```
for t in range(len(test)):
    model = ARIMA(history, order=(5,1,0))
    model_fit = model.fit(disp=0)
    output = model_fit.forecast()
    yhat = output[0]
    predictions.append(yhat)
    obs = test[t]
    history.append(obs)
    print('predicted=%f, expected=%f' % (yhat, obs))
```

```
predicted=239.755179, expected=227.000000
predicted=220.737300, expected=234.000000
predicted=237.815008, expected=264.000000
predicted=252.750592, expected=302.000000
predicted=306.715794, expected=293.000000
predicted=285.374643, expected=259.000000
predicted=250.264004, expected=229.000000
predicted=227.093124, expected=203.000000
predicted=211.011455, expected=229.000000
predicted=253.260281, expected=242.000000
predicted=252.490682, expected=233.000000
predicted=234.042132, expected=267.000000
predicted=268.773632, expected=269.000000
predicted=261.782257, expected=270.000000
predicted=271.798054, expected=315.000000
predicted=314.422115, expected=364.000000
predicted=368.637742, expected=347.000000
predicted=334.957878, expected=312.000000
predicted=301.161818, expected=274.000000
predicted=265.936481, expected=237.000000
predicted=244.037181, expected=278.000000
predicted=312.961792, expected=284.000000
predicted=291.748167, expected=277.000000
predicted=284.551868, expected=317.000000
predicted=316.501195, expected=313.000000
predicted=303.218148, expected=318.000000
predicted=324.834619, expected=374.000000
predicted=373.140656, expected=413.000000
predicted=415.007200, expected=405.000000
predicted=397.508453, expected=355.000000
predicted=332.087112, expected=306.000000
predicted=299.452956, expected=271.000000
predicted=279.908341, expected=306.000000
```

```
error = mean_squared_error(test, predictions)
print('Test MSE: %.3f' % error)
```

Test MSE: 782.495

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



Comparing Multiple Models

```
df2=df1.copy()
```

```
df2.head()
```

Month	Passengers
1995-01-01	112
1995-02-01	118
1995-03-01	132
1995-04-01	129
1995-05-01	121

```
df2['Month']=pd.DatetimeIndex(df['Month']).month
```

```
df2.head()
```

Month	Passengers	Month
1995-01-01	112	1
1995-02-01	118	2
1995-03-01	132	3
1995-04-01	129	4
1995-05-01	121	5

```
df2 = pd.get_dummies(data=df2, columns =['Month'])
```

```
df2.head()
```

Month	Passengers	Month_1	Month_2	Month_3	Month_4	Month_5	Month_6	Month_7	Month_8	Month_9	Month_10	Month_11	Month_12
1995-01-01	112	1	0	0	0	0	0	0	0	0	0	0	0
1995-02-01	118	0	1	0	0	0	0	0	0	0	0	0	0
1995-03-01	132	0	0	1	0	0	0	0	0	0	0	0	0
1995-04-01	129	0	0	0	1	0	0	0	0	0	0	0	0
1995-05-01	121	0	0	0	0	1	0	0	0	0	0	0	0

```
df2.shape
```

(96, 13)

```
t= np.arange(1,97)
```

```
df2['t']=t
```

```
df2['t_sq'] = df2['t']*df2['t']
```

```
log_Passengers=np.log(df2['Passengers'])
```

```
df2['log_Passengers']=log_Passengers
```

```
df2.head()
```

Month	Passengers	Month_1	Month_2	Month_3	Month_4	Month_5	Month_6	Month_7	Month_8	Month_9	Month_10	Month_11	Month_12	t	t_sq	log_Passengers
1995-01-01	112	1	0	0	0	0	0	0	0	0	0	0	0	1	1	4.718499
1995-02-01	118	0	1	0	0	0	0	0	0	0	0	0	0	2	4	4.770685
1995-03-01	132	0	0	1	0	0	0	0	0	0	0	0	0	3	9	4.882802
1995-04-01	129	0	0	0	1	0	0	0	0	0	0	0	0	4	16	4.859812
1995-05-01	121	0	0	0	0	1	0	0	0	0	0	0	0	5	25	4.795791

```
train1, test1 = np.split(df2, [int(.67 *len(df2))])
```

```
linear= smf.ols('Passengers ~ t',data=train1).fit()
predlin=pd.Series(linear.predict(pd.DataFrame(test1['t'])))
rmselin=np.sqrt((np.mean(np.array(test1['Passengers'])-np.array(predlin))**2))
rmselin
```

25.503983516483537

```
quad=smf.ols('Passengers~t+t_sq',data=train1).fit()
predquad=pd.Series(quad.predict(pd.DataFrame(test1[['t','t_sq']])))
rmsequad=np.sqrt(np.mean((np.array(test1['Passengers'])-np.array(predquad))**2))
rmsequad
```

53.18955514415421

```
expo=smf.ols('Passengers~t',data=train1).fit()
predexp=pd.Series(expo.predict(pd.DataFrame(test1['t'])))
rmseexpo=np.sqrt(np.mean((np.array(test1['Passengers'])-np.array(np.exp(predexp)))**2))
rmseexpo
```

1.603094593327857e+128

```
additive= smf.ols('Passengers~ Month_1+Month_2+Month_3+Month_4+Month_5+Month_6+Month_7+Month_8+Month_9+Month_10+Month_11+Month_12',data=train1).fit()
predadd=pd.Series(additive.predict(pd.DataFrame(test1[['Month_1','Month_2','Month_3','Month_4','Month_5','Month_6','Month_7','Month_8','Month_9','Month_10','M
rmseadd=np.sqrt(np.mean((np.array(test1['Passengers'])-np.array(predadd))**2))
rmseadd
```

123.94935401651406

```
addlinear= smf.ols('Passengers~t+Month_1+Month_2+Month_3+Month_4+Month_5+Month_6+Month_7+Month_8+Month_9+Month_10+Month_11+Month_12',data=train1).fit()
predaddlinear=pd.Series(addlinear.predict(pd.DataFrame(test1[['t','Month_1','Month_2','Month_3','Month_4','Month_5','Month_6','Month_7','Month_8','Month_9','M
rmseaddlinear=np.sqrt(np.mean((np.array(test1['Passengers'])-np.array(predaddlinear))**2))
rmseaddlinear
```

34.26029387076406

```
mulsea=smf.ols('log_Passengers~Month_1+Month_2+Month_3+Month_4+Month_5+Month_6+Month_7+Month_8+Month_9+Month_10+Month_11+Month_12',data=train1).fit()
predmul= pd.Series(mulsea.predict(pd.DataFrame(test1[['Month_1','Month_2','Month_3','Month_4','Month_5','Month_6','Month_7','Month_8','Month_9','Month_10','Mo
rmsemul= np.sqrt(np.mean((np.array(test1['Passengers'])-np.array(np.exp(predmul)))**2))
rmsemul
```

127.83693479585908

```
mullin= smf.ols('log_Passengers~t+Month_1+Month_2+Month_3+Month_4+Month_5+Month_6+Month_7+Month_8+Month_9+Month_10+Month_11+Month_12',data=train1).fit()
predmullin= pd.Series(mullin.predict(pd.DataFrame(test1[['t','Month_1','Month_2','Month_3','Month_4','Month_5','Month_6','Month_7','Month_8','Month_9','Month_
rmsemulin=np.sqrt(np.mean((np.array(test1['Passengers'])-np.array(np.exp(predmullin)))**2))
rmsemulin
```

12.759135120295419

```
mul_quad= smf.ols('log_Passengers~t+t_sq+Month_1+Month_2+Month_3+Month_4+Month_5+Month_6+Month_7+Month_8+Month_9+Month_10+Month_11+Month_12',data=train1).fit(
pred_mul_quad= pd.Series(mul_quad.predict(test1[['t','t_sq','Month_1','Month_2','Month_3','Month_4','Month_5','Month_6','Month_7','Month_8','Month_9','Month_1
rmse_mul_quad=np.sqrt(np.mean((np.array(test1['Passengers'])-np.array(np.exp(pred_mul_quad)))**2))
rmse_mul_quad
```

28.25957879076086

```
output = {'Model':pd.Series(['rmse_mul_quad','rmseadd','rmseaddlinear','rmseaddquad','rmseexpo','rmselin','rmsemul','rmsemulin','rmsequad']),
          'Values':pd.Series([rmse_mul_quad,rmseadd,rmseaddlinear,rmseexpo,rmselin,rmsemul,rmsemulin,rmsequad])}
```

```
rmse=pd.DataFrame(output)
```

```
print(rmse)
```

```
           Model         Values
0  rmse_mul_quad   2.825958e+01
1        rmseadd   1.239494e+02
2  rmseaddlinear   3.426029e+01
3    rmseaddquad  1.603095e+128
4       rmseexpo   2.550398e+01
5        rmselin   1.278369e+02
6        rmsemul   1.275914e+01
7      rmsemulin   5.318956e+01
8       rmsequad            NaN
```