In [186…

```
#####################
# Dependency import #
#####################

import seaborn as sb
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sklearn.linear_model as linear_model
import sklearn.preprocessing as preprocessing
import sklearn.model_selection as model_selection
import sklearn.metrics as metrics
import json
```

In [187…

```
###################
# Dataset loading #
###################

# Dataset: https://www.kaggle.com/datasets/sudalairajkumar/daily-temperature-of-maj
dt = pd.read_csv('city_temperature.csv')
dt.head(100)
```

```
C:\Users\teodo\AppData\Local\Temp\ipykernel_3184\1848890338.py:6: DtypeWarning: Colu
mns (2) have mixed types. Specify dtype option on import or set low_memory=False.
  dt = pd.read_csv('city_temperature.csv')
```

Out[187…

	Region	Country	State	City	Month	Day	Year	AvgTemperature
0	Africa	Algeria	NaN	Algiers	1	1	1995	64.2
1	Africa	Algeria	NaN	Algiers	1	2	1995	49.4
2	Africa	Algeria	NaN	Algiers	1	3	1995	48.8
3	Africa	Algeria	NaN	Algiers	1	4	1995	46.4
4	Africa	Algeria	NaN	Algiers	1	5	1995	47.9
...	...	...	...	...	...	...	...	...
95	Africa	Algeria	NaN	Algiers	4	6	1995	59.0
96	Africa	Algeria	NaN	Algiers	4	7	1995	54.9
97	Africa	Algeria	NaN	Algiers	4	8	1995	54.2
98	Africa	Algeria	NaN	Algiers	4	9	1995	57.8
99	Africa	Algeria	NaN	Algiers	4	10	1995	60.0

100 rows × 8 columns

In [188…

```
#################
# Data cleaning #
#################
```

```python
dt.drop(columns=["State", "Region"], axis=1, inplace=True)
dt.drop(index=dt.loc[dt["Day"] <= 0].index, inplace=True)
dt.drop(index=dt.loc[dt["AvgTemperature"] <= -20].index, inplace=True)
dt.dropna()

dt
```

Out[188…

	Country	City	Month	Day	Year	AvgTemperature
0	Algeria	Algiers	1	1	1995	64.2
1	Algeria	Algiers	1	2	1995	49.4
2	Algeria	Algiers	1	3	1995	48.8
3	Algeria	Algiers	1	4	1995	46.4
4	Algeria	Algiers	1	5	1995	47.9
...	...	...	...	...	...	...
2906322	US	San Juan Puerto Rico	7	27	2013	82.4
2906323	US	San Juan Puerto Rico	7	28	2013	81.6
2906324	US	San Juan Puerto Rico	7	29	2013	84.2
2906325	US	San Juan Puerto Rico	7	30	2013	83.8
2906326	US	San Juan Puerto Rico	7	31	2013	83.6

2825666 rows × 6 columns

In [189…

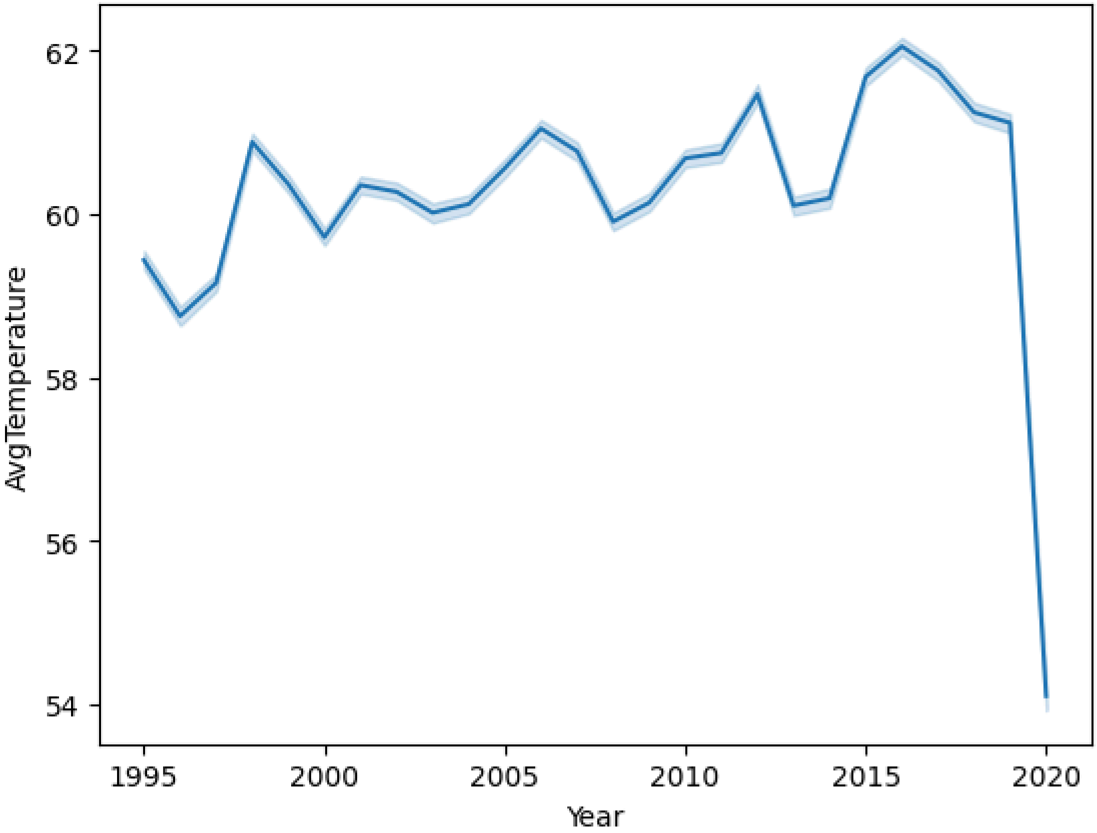
```python
############################################
# MSE and R2 progression datasets assembly #
############################################

index = 0
g_mse_comp = {"Training MSE" : [], "Test MSE": []}
g_mse_comp_indexes = []
g_r2_progression = {"Polynomial degree" : [], "R2": []}
g_mse_progression = {"Polynomial degree" : [], "MSE": []}
g_model = None


l_mse_comp = {"Training MSE" : [], "Test MSE": []}
l_mse_comp_indexes = []
l_r2_progression = {"Polynomial degree" : [], "R2": []}
l_mse_progression = {"Polynomial degree" : [], "MSE": []}
l_model = None


def addMetrics(t_mse, c_mse, r2, i, mse_comp, mse_comp_indexes, r2_progression, mse
    print("\n\n")
    print("Polynomial degree:", i)
    print("Training MSE:", t_mse)
    print("MSE:", c_mse)
    print("R2:", r2)

    mse_comp["Training MSE"].append(t_mse)
    mse_comp["Test MSE"].append(c_mse)
    mse_comp_indexes.append(len(mse_comp["Test MSE"]))

    r2_progression["Polynomial degree"].append(i)
    r2_progression["R2"].append(r2)

    mse_progression["Polynomial degree"].append(i)
    mse_progression["MSE"].append(c_mse)
```

In [190…

```
########################################
# Visualisation of global temperature #
########################################
sb.lineplot(data=dt, x="Year", y="AvgTemperature")
```

Out[190…

```
<Axes: xlabel='Year', ylabel='AvgTemperature'>
```



In [191…

```
global g_mse_comp
global g_mse_comp_indexes
global g_r2_progression
global g_mse_progression
global index
global g_model
index = 1

tmp = dt.groupby(by=["Day", "Month", "Year"])["AvgTemperature"].mean().reset_index(

mse = None
```

```
r2 = None

degree = 1


for i in range(1, 100):
    poly_features = preprocessing.PolynomialFeatures(degree=i, include_bias=False).
    x_train, x_test, y_train, y_test = model_selection.train_test_split(poly_featur

    _model = linear_model.LinearRegression()
    poly_regression = _model.fit(x_train, y_train)

    predictions = poly_regression.predict(x_test)

    t_predictions = poly_regression.predict(x_train)

    t_mse = metrics.mean_squared_error(y_pred=t_predictions, y_true=y_train)
    c_mse = metrics.mean_squared_error(y_pred=predictions, y_true=y_test)
    c_r2 = metrics.r2_score(y_pred=predictions, y_true=y_test)


    # If the MSE calculated on the training data is smaller
    # than the one calculated on the testing data is smaller
    # then the model is overfitted and the operation stops
    if mse is not None and r2 is not None:
        if c_mse > t_mse:
            if c_mse <= mse or c_r2 >= r2:
                g_model = _model
                degree = i
            else:
                break
    else:
        g_model = _model

    addMetrics(t_mse, c_mse, r2, i, g_mse_comp, g_mse_comp_indexes, g_r2_progressio
    mse = c_mse
    r2 = c_r2


g_model = {"x_intercept": g_model.intercept_, "Beta_Coefficients":  g_model.coef_.t
model_file = open(file="global_temp_model.json", mode="w")
jf = json.dump(obj=g_model,fp= model_file, indent=4)
model_file.flush()
model_file.close()
del(model_file)


g_comp = pd.DataFrame(data=g_mse_comp, index=g_mse_comp_indexes)
g_r2_prog = pd.DataFrame(data=g_r2_progression, index=g_mse_comp_indexes)
g_mse_prog = pd.DataFrame(data=g_mse_progression, index=g_mse_comp_indexes)

g_comp
```

```
Polynomial degree: 1
Training MSE: 121.80854926740395
MSE: 121.95943113334678
R2: None



Polynomial degree: 2
Training MSE: 16.712176274427712
MSE: 17.09688391265149
R2: 0.05159481356687867



Polynomial degree: 3
Training MSE: 11.765339747714172
MSE: 11.22281746074128
R2: 0.8672040839042617



Polynomial degree: 4
Training MSE: 5.917898110822623
MSE: 6.3669420620092545
R2: 0.9128490838141019



Polynomial degree: 5
Training MSE: 6.055017848913137
MSE: 6.05986559709399
R2: 0.9511416505104127



Polynomial degree: 6
Training MSE: 6.140369433910765
MSE: 5.864773134272614
R2: 0.9523371282815597



Polynomial degree: 7
Training MSE: 6.131243349666638
MSE: 5.894737272906322
R2: 0.9538419762031933
```

Out[191…

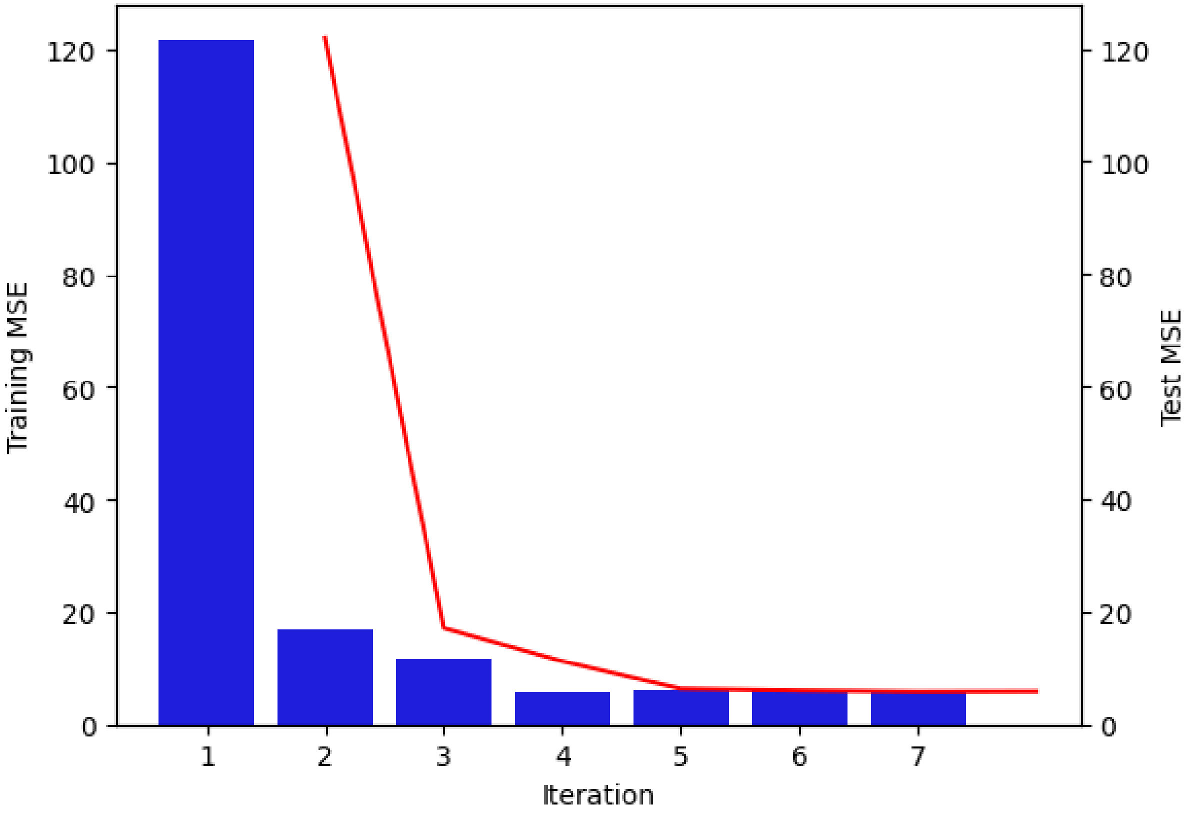
	Training MSE	Test MSE
1	121.808549	121.959431
2	16.712176	17.096884
3	11.765340	11.222817
4	5.917898	6.366942
5	6.055018	6.059866
6	6.140369	5.864773
7	6.131243	5.894737

In [192…

```
##################################################################################
# Comparison of the MSE in predicting training data VS testing data for the Global
##################################################################################
p1 = sb.barplot(data=g_comp, x=g_comp.index, y="Training MSE", color="b")
p1.set_xlabel("Iteration")
p1.set_ylim(bottom=0)
plt.twinx()
p2 = sb.lineplot(data=g_comp, x=g_comp.index, y="Test MSE",color="r")
p2.set_xlabel("Iteration")
p2.set_ylim(bottom=0)
```

Out[192…

```
(0.0, 127.76416403330049)
```

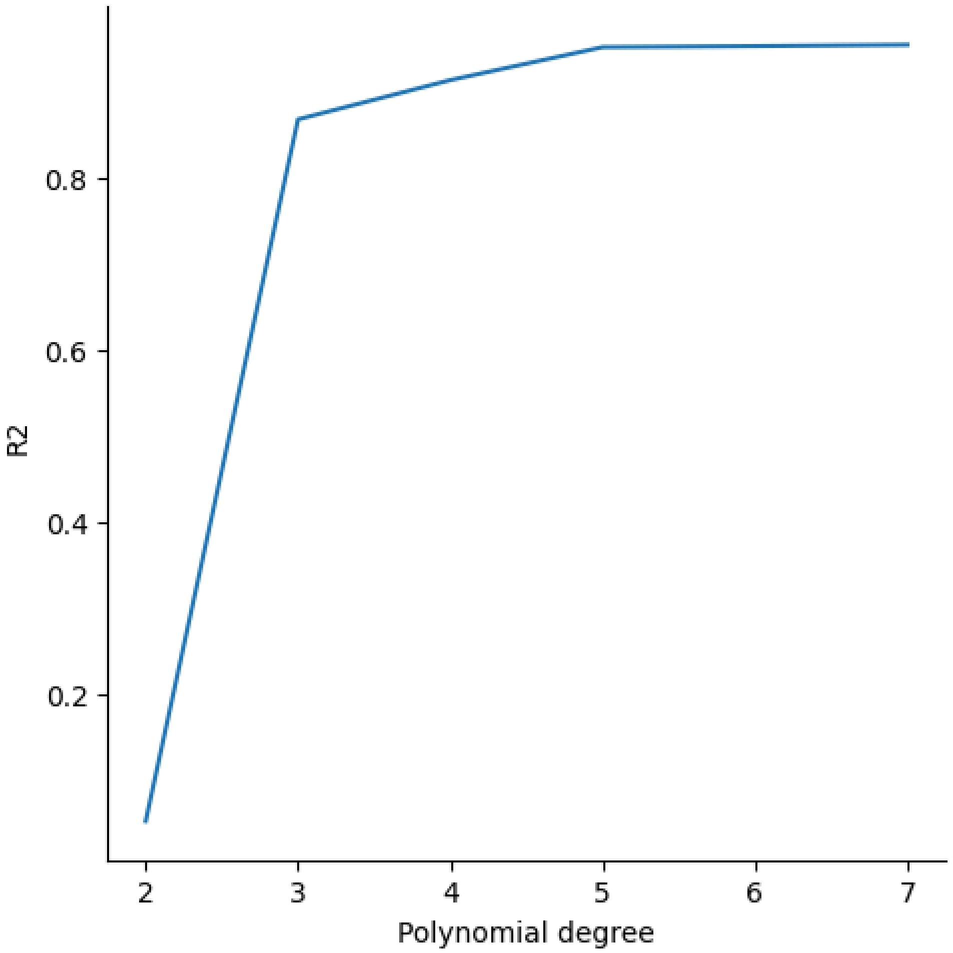


In [193…

```
##################################################################################
# Visualisation of the R2 score progression in relation with the polynomial degree
```

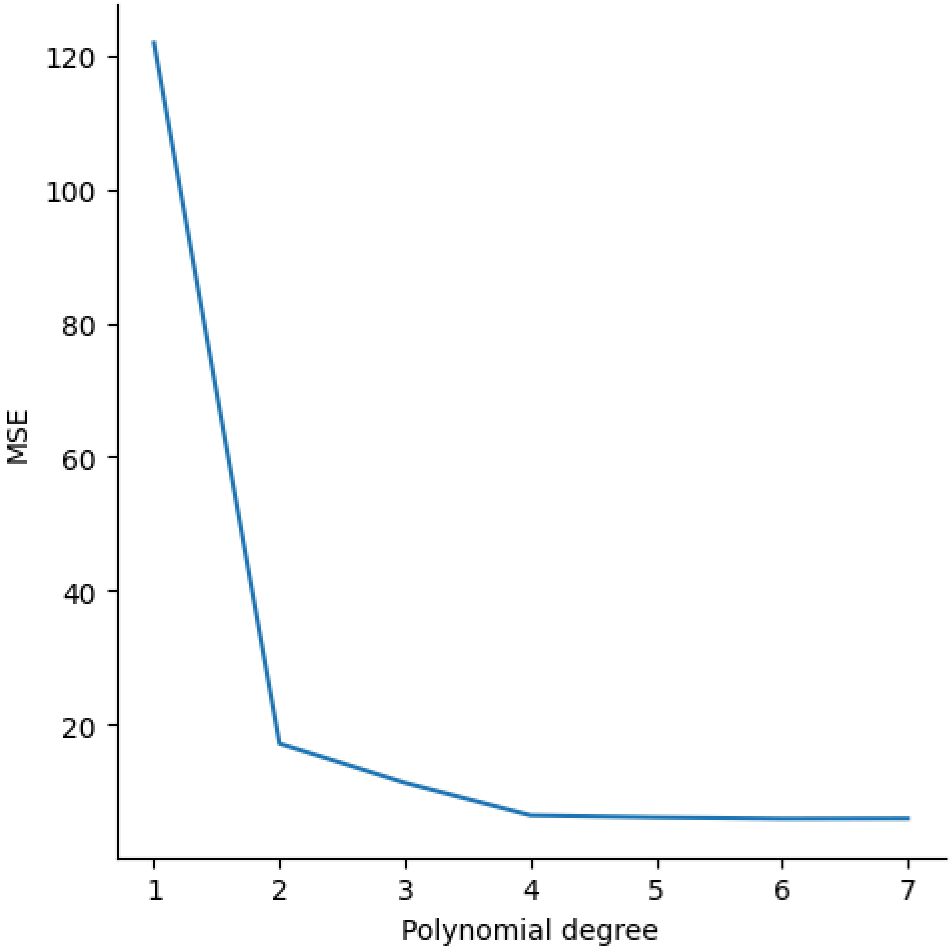
```
###################################################################################

sb.relplot(data= g_r2_prog, x="Polynomial degree", y="R2", kind="line")
```

```
Out[193…  <seaborn.axisgrid.FacetGrid at 0x213f23d1700>
```



```
In [194…
###################################################################################
# Visualisation of the MSE score progression in relation with the polynomial degree
###################################################################################

sb.relplot(data= g_mse_prog, x="Polynomial degree", y="MSE", kind="line")
```

```
Out[194…  <seaborn.axisgrid.FacetGrid at 0x213f2420830>
```

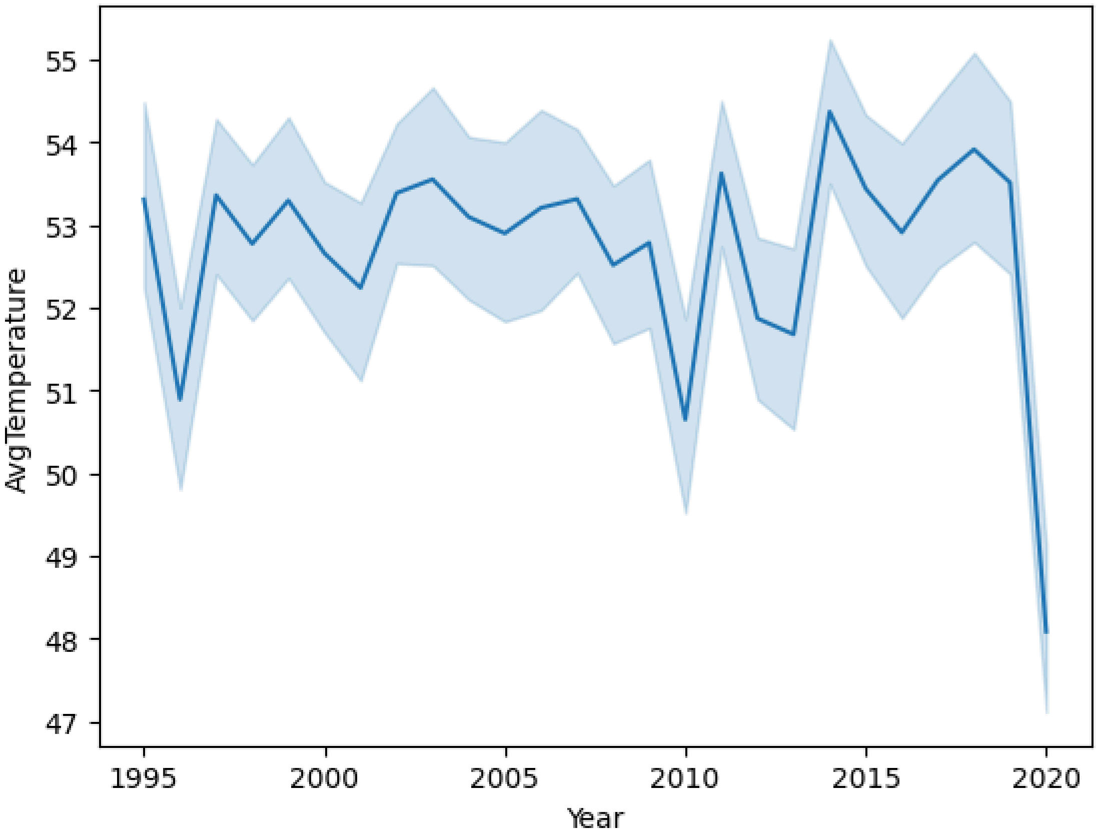


In [195…

```
#########################################
# Visualisation of London's temperature #
#########################################

dtl =  dt[(dt['Country'] == 'United Kingdom') & (dt['City'] == 'London')]
sb.lineplot(data=dtl, x="Year", y="AvgTemperature")
```

Out[195…

```
<Axes: xlabel='Year', ylabel='AvgTemperature'>
```



In [196…

```
global l_mse_comp
global l_mse_comp_indexes
global l_r2_progression
global l_mse_progression
global index
global l_model
index = 1

mse = None
r2 = None

degree = 1
model = None

london_temp_model = None


for i in range(1, 100):
    poly_features = preprocessing.PolynomialFeatures(degree=i, include_bias=False).
    x_train, x_test, y_train, y_test = model_selection.train_test_split(poly_featur

    _model = linear_model.LinearRegression()
    poly_regression = _model.fit(x_train, y_train)

    predictions = poly_regression.predict(x_test)

    t_predictions = poly_regression.predict(x_train)

    t_mse = metrics.mean_squared_error(y_pred=t_predictions, y_true=y_train)
    c_mse = metrics.mean_squared_error(y_pred=predictions, y_true=y_test)
    c_r2 = metrics.r2_score(y_pred=predictions, y_true=y_test)


    # If the MSE calculated on the training data is smaller
    # than the one calculated on the testing data is smaller
    # then the model is overfitted and the operation stops
    if mse is not None and r2 is not None:
        if c_mse > t_mse:
            if c_mse <= mse or c_r2 >= r2:
                l_model = _model
                degree = i
            else:
                break
    else:
        l_model = _model
    addMetrics(t_mse, c_mse, r2, i, l_mse_comp, l_mse_comp_indexes, l_r2_progressio
    mse = c_mse
    r2 = c_r2


l_model = {"x_intercept": l_model.intercept_, "Beta_Coefficients":  l_model.coef_.t
model_file = open(file="london_temp_model.json", mode="w")
jf = json.dump(obj=l_model,fp= model_file, indent=4)
model_file.flush()
model_file.close()
del(model_file)


l_comp = pd.DataFrame(data=l_mse_comp, index=l_mse_comp_indexes)
l_r2_prog = pd.DataFrame(data=l_r2_progression, index=l_mse_comp_indexes)
l_mse_prog = pd.DataFrame(data=l_mse_progression, index=l_mse_comp_indexes)

l_comp
```

```
Polynomial degree: 1
Training MSE: 93.02099949960763
MSE: 98.13971467118246
R2: None



Polynomial degree: 2
Training MSE: 38.09977344195464
MSE: 39.940284542694066
R2: 0.06189908362279428



Polynomial degree: 3
Training MSE: 31.527136203876474
MSE: 34.174856840690424
R2: 0.596832968977231



Polynomial degree: 4
Training MSE: 26.629754597072782
MSE: 28.171747381736626
R2: 0.6652071566868334



Polynomial degree: 5
Training MSE: 27.021416040455524
MSE: 27.264118185339864
R2: 0.7250730073107596
```

Out[196…

	Training MSE	Test MSE
1	93.020999	98.139715
2	38.099773	39.940285
3	31.527136	34.174857
4	26.629755	28.171747
5	27.021416	27.264118

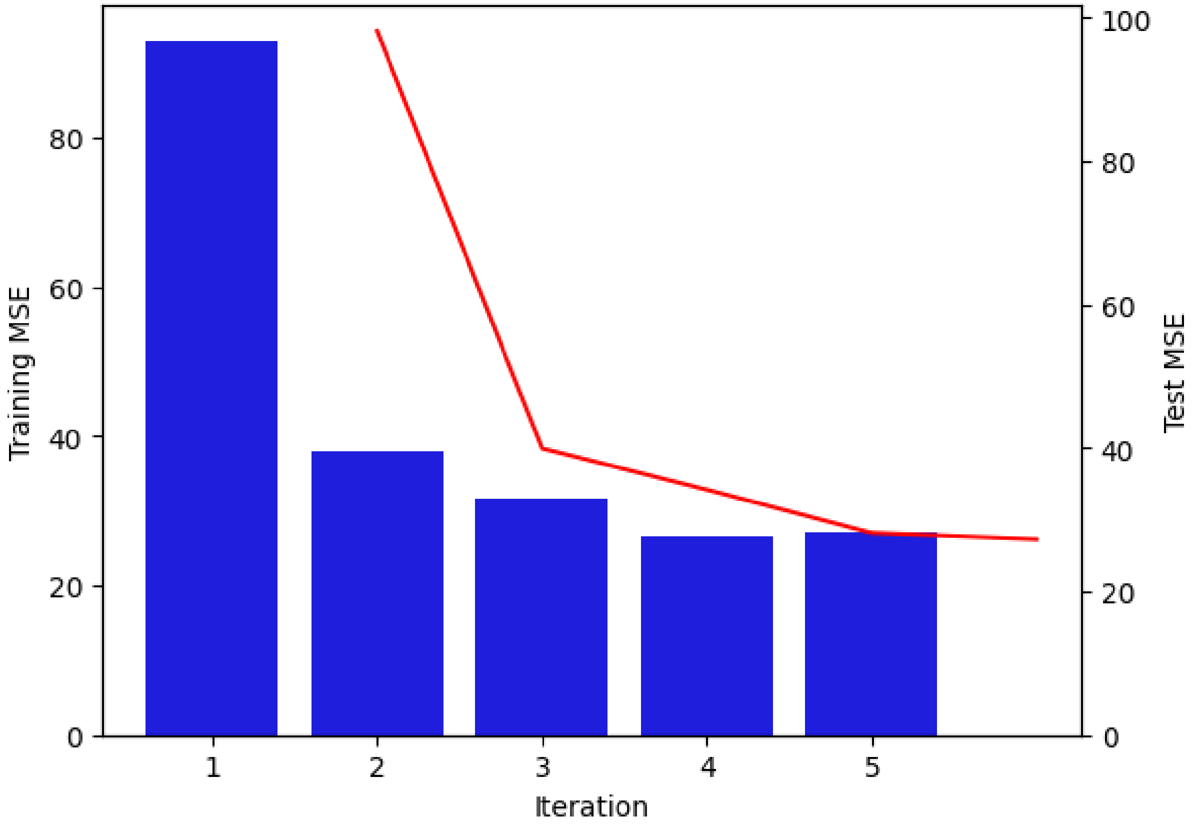
In [197…

```
##################################################################################
# Comparison of the MSE in predicting training data VS testing data for the Global
##################################################################################
p1 = sb.barplot(data=l_comp, x=l_comp.index, y="Training MSE", color="b")
p1.set_xlabel("Iteration")
p1.set_ylim(bottom=0)
plt.twinx()
p2 = sb.lineplot(data=l_comp, x=l_comp.index, y="Test MSE",color="r")
```

```
p2.set_xlabel("Iteration")
p2.set_ylim(bottom=0)
```

Out[197…

```
(0.0, 101.68349449547459)
```

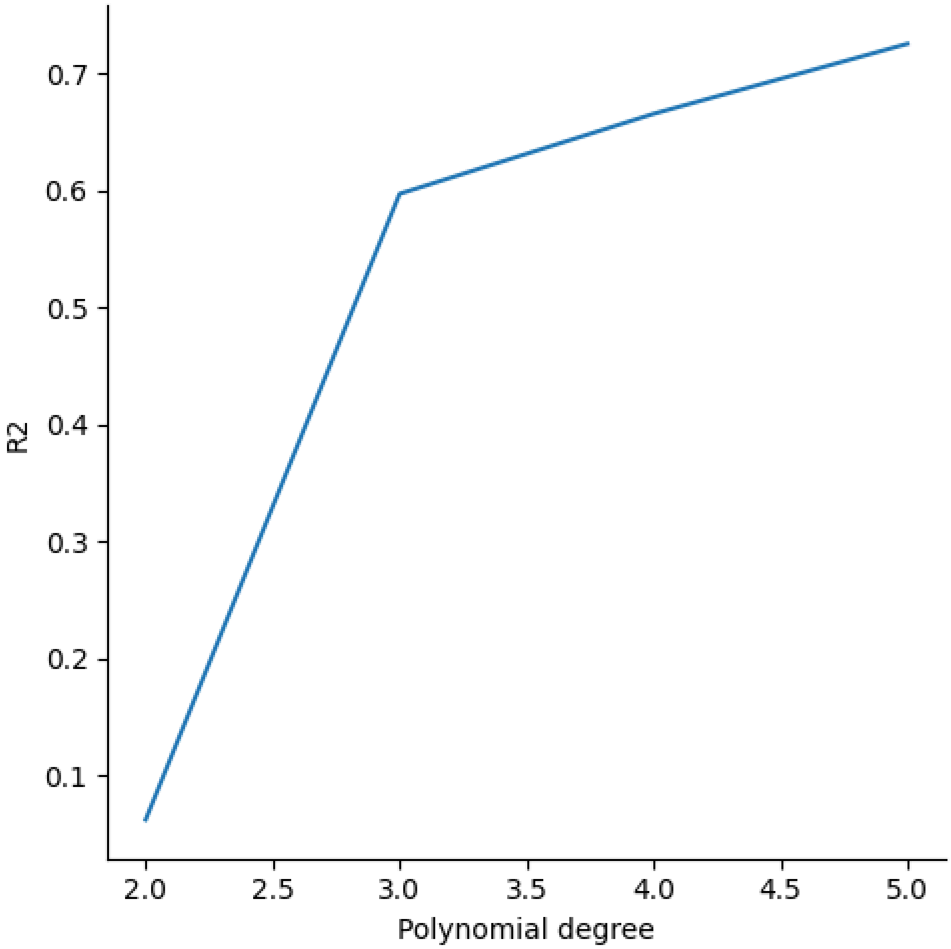


In [198…

```
###################################################################################
# Visualisation of the R2 score progression in relation with the polynomial degree
###################################################################################

sb.relplot(data= l_r2_prog, x="Polynomial degree", y="R2", kind="line")
```

Out[198…

```
<seaborn.axisgrid.FacetGrid at 0x213f25fb9e0>
```

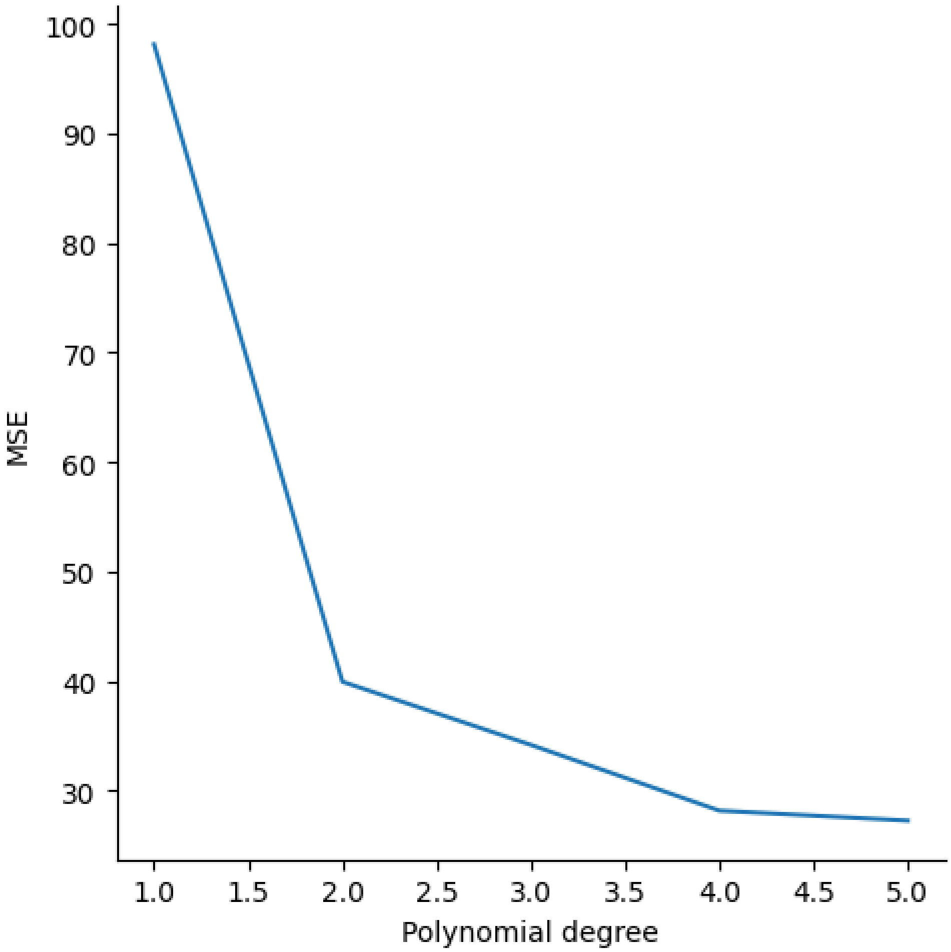


In [199…

```
#################################################################################
# Visualisation of the MSE score progression in relation with the polynomial degree
#################################################################################

sb.relplot(data= l_mse_prog, x="Polynomial degree", y="MSE", kind="line")
```

Out[199…

```
<seaborn.axisgrid.FacetGrid at 0x213bf6e77d0>
```


100
90
80
70
60
50
40
30
MSE
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0
Polynomial degree