Using Polynomial Regression to Predict Coronal Index based on Sunspot Number

Introduction

The corona is the thin outer layer of the Sun, extending for millions of kilometres and merging into a stream of high-energy particles moving away from the Sun known as the solar wind. More energetic explosions such as solar flares and coronal mass ejections also originate from the corona, and these can have profound effects on Earth, damaging power grids, satellites, and other devices [1]. The corona is so thinly dispersed that it can only be seen during a total eclipse or with special telescopes, but its activity can be measured with the coronal index, which indicates the irradiance emitted by the corona [2]. Sunspots, on the other hand, can be seen even without a telescope. They appear as temporary darker spots on the Sun's surface and act as a measure of the Sun's activity. They are caused by magnetic plasma rising to the Sun's surface, transferring energy from the surface into the corona [1].

Machine learning is applied to predict the daily coronal index based on the number of sunspots, which would be useful for understanding the Sun's activity and predicting possibly harmful effects on Earth. The report will first discuss the problem formulation, then data processing, methods, results, and lastly conclusions and improvements. The code is provided as an appendix.

Problem Formulation

The data points for this problem are days. Sunspot number is the feature as it is relatively easy to compute, and coronal index is the label as obtaining it often requires more elaborate techniques. The sunspot number is an integer ranging between 0 and 500 and the coronal index is a number usually ranging between 0 and 25. The data is obtained from the SILSO website [2] and NOAA [3].

Methods

The number of data points is limited by the coronal index data, which is available from 1939 to 2008, giving a total of 25567 datapoints. There is no missing data within this time period. In addition to sunspot number, the sunspot data contains columns for fractional date, standard deviation, number of observations, and definitive/provisional indicator, which are dropped at the preprocessing stage. There are no additional columns in the coronal index data set. Both data sets contain columns for year, month, and day, which are used to create a date column so that the data sets can be merged. The sunspot number was selected as the feature because it is a well-known indicator for solar activity and the data set was readily available. The coronal index should also correlate with it. Other similar features such as sunspot area could also be used but the data might be harder to obtain.

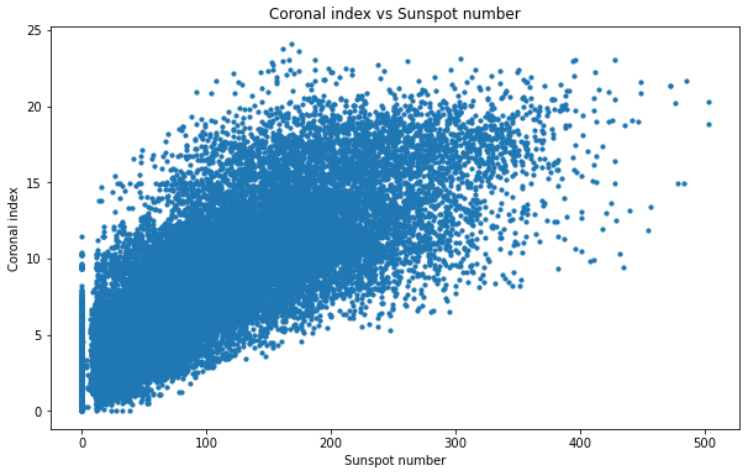


Figure 1. Data visualization.

Based on the graph of the coronal index versus sunspot number in Figure 1 and the fact that both are measures of the Sun's activity, it seems that the relationship between the coronal index and sunspot number might be polynomial. Thus, a natural choice for the hypothesis space would be polynomial maps with degrees ranging for example from 1 to 6 as the relationship does not seem to be extremely non-linear. However, the graph also shows that in many cases several different label values correspond to a single feature value, which is likely to decrease the prediction accuracy. Thus, previous days' sunspot numbers will be used to construct another model with five features. Using the previous four days' values reduces the number of data points to 25563.

Thus, polynomial regression with one feature and five features and degrees from 1 to 6 will be analysed and compared. In both cases the loss function is the squared error loss because it is often used with polynomial regression methods. The same loss function will be used to compute the training, validation, and test errors. The validation error will be used for comparing the models and test error for evaluating the overall performance of the models. The data will be split into training, validation, and test sets using the train_test_split -function from Scikit-learn with sizes 60%, 20%, and 20% because it is a common split ratio and ensures that most data is used for training but enough data still remains for evaluating the model with the validation and test errors.

The hypothesis space for polynomial regression consists of polynomial maps of the form

$$h^{(\boldsymbol{w})}x = \sum_{j=1}^{n} w_j x^{j-1},$$

with feature value x and parameter vector $\boldsymbol{w} = (w_1, \dots, w_n)^T \in \mathbb{R}^n$ [4]. These are fitted to the data by minimizing the average squared error loss

$$\frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - h^{(\boldsymbol{w})}(x^{(i)}))^2,$$

where y is the actual label value and $h^{(\boldsymbol{w})}(x)$ is the predicted label value [5]. In practice this is done by first transforming the feature vector and then applying linear regression.

Results

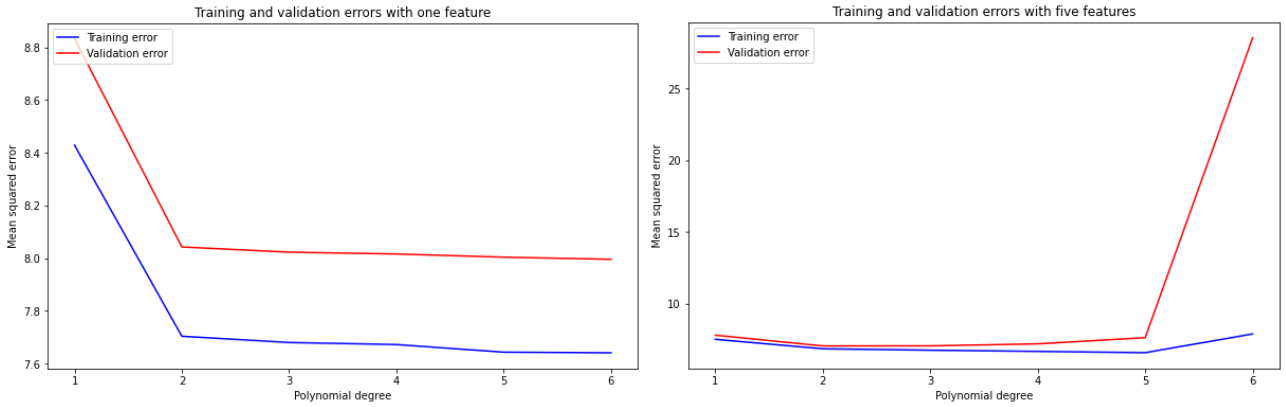


Figure 2. Comparison of training and validation errors.

Table 1. Training and validation errors.

Polynomial degree	One feature		Five features	
	Training error	Validation error	Training error	Validation error
1	8.428827	8.833335	7.515352	7.804029
2	7.703289	8.042481	6.852728	7.053289
3	7.679684	8.022726	6.754261	7.058372
4	7.671935	8.015845	6.670343	7.198424
5	7.642963	8.003734	6.580968	7.630091
6	7.640522	7.995622	7.888221	28.558834

As can be seen from Figure 2 and Table 1, the training and validation errors are quite similar for both polynomial regression models, which seems reasonable with such a large number of data points. However, the errors still seem somewhat large for such a relatively small range of label values although there is no baseline error for reliable comparison. There is perhaps some underfitting especially when using only one feature, which is also supported by the data visualization in Figure 1 as several different label values are associated with a single feature value. Using five features clearly decreases both the training and validations errors but it is likely that the accuracy could still be improved with better features. Some overfitting can also be seen for example in the case of five features and polynomial degree 6, where the validation error is much larger than the training error. Based on the validation errors in Table 1, the best regression model is with degree 6 with one feature and 2 with five features. Overall the best results seem to be obtained with five features and thus the chosen model is polynomial regression with five features and degree 2 as it has the lowest validation error. Test error of this final chosen method is 7.073610, which is in line with the other errors.

Conclusion

The aim of this report was to apply polynomial regression to predict daily coronal index based on sunspot number. Two polynomial regression models with different number of features were trained and the resulting training, validation, and test errors analysed and compared. The validation and test errors suggest that best accuracy is obtained using five features and polynomial regression of degree 2, for which the final test error was 7.073610. The results, however, could likely still be improved as the errors were relatively large compared to the range of label values. It seems that the models were mostly limited by the weak correlation between the feature and label values. Still, regularization in the form of ridge regression, for example, could be applied to reduce the effect of outliers in the data as well as possible overfitting even though it does not seem to be that big of an issue even with the used polynomial regression models. It might also be interesting to see whether more complex models such as neural networks could be able to better capture some underlaying relationships in the data.

References

[1] H. Zirin and K. Lang, “Sun”, Encyclopedia Britannica, 2021, https://www.britannica.com/place/Sun.

[2] National Oceanic and Atmospheric Administration, “Readme: Coronal Index of Solar Activity”, 2013, https://www.ngdc.noaa.gov/stp/space-weather/solar-data/solar-indices/solar_corona/documentation/readme_solar-corona.pdf.

[3] World Data Center, Sunspot Number and Long-term Solar Observations, Royal Observatory of Belgium, Brussels, https://wwwbis.sidc.be/silso/datafiles.

[4] National Oceanic and Atmospheric Administration, Coronal Index, https://www.ngdc.noaa.gov/stp/space-weather/solar-data/solar-indices/solar_corona/coronal-index/slovak/tables/.

[5] A. Jung, “Machine Learning: The Basics”, Springer, Singapore, 2022.

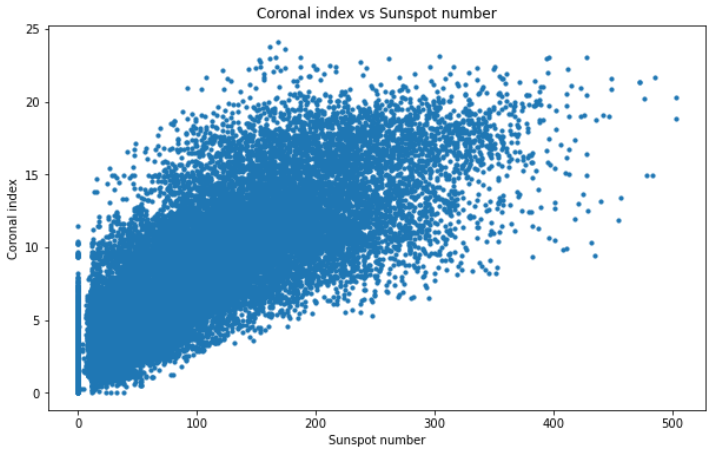
Appendix

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
```

In [2]:
```
df_1 = pd.read_csv('sunspot_number.csv', sep = ';', names = ['Year', 'Month', 'Day', 'Frac', 'Sunspot number', 'Std', 'Obs', 'Ind'], header = None)

data_1 = df_1.assign(Date = df_1["Year"].astype(str) + '-' + df_1["Month"].astype(str) + '-' + df_1["Day"].astype(str))
data_1 = data_1.drop(['Year', 'Month', 'Day', 'Frac', 'Std', 'Obs', 'Ind'], axis=1)
data_1 = data_1[['Date', 'Sunspot number']]
```

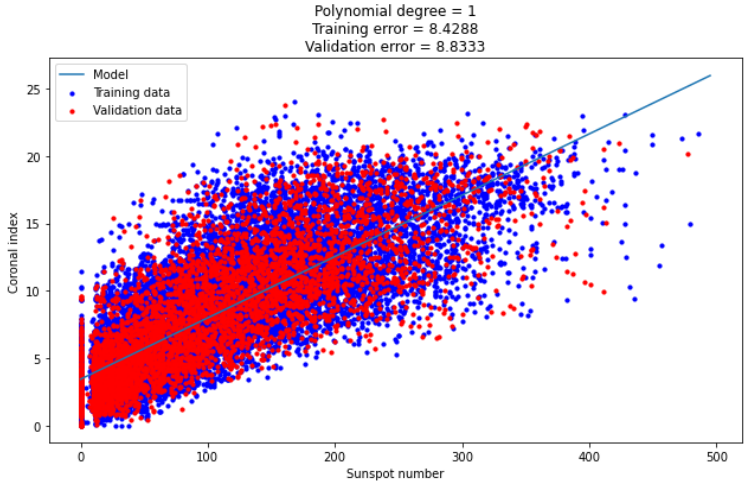
In [3]:
```
df_2 = pd.read_csv('coronal_index.txt', delim_whitespace = True, names = ['Year', 'Month', 'Day', 'Coronal index'], header = None)

data_2 = df_2.assign(Date = df_2["Year"].astype(str) + '-' + df_2["Month"].astype(str) + '-' + df_2["Day"].astype(str))
data_2 = data_2.drop(['Year', 'Month', 'Day'], axis=1)
data_2 = data_2[['Date', 'Coronal index']]
```

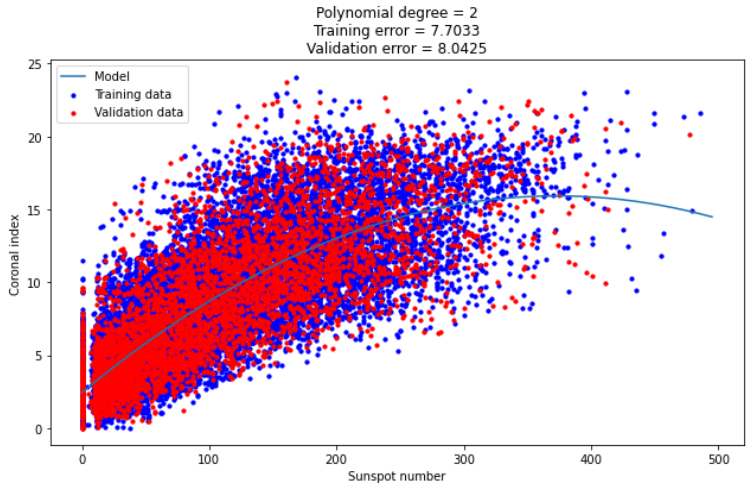
In [4]:
```
data = pd.merge(data_1, data_2, on = 'Date')

plt.figure(figsize = (10, 6))

plt.scatter(data['Sunspot number'], data['Coronal index'], s = 10)
plt.xlabel('Sunspot number', size = 10)
plt.ylabel('Coronal index', size = 10)
plt.title('Coronal index vs Sunspot number', size = 12)

plt.show()
```

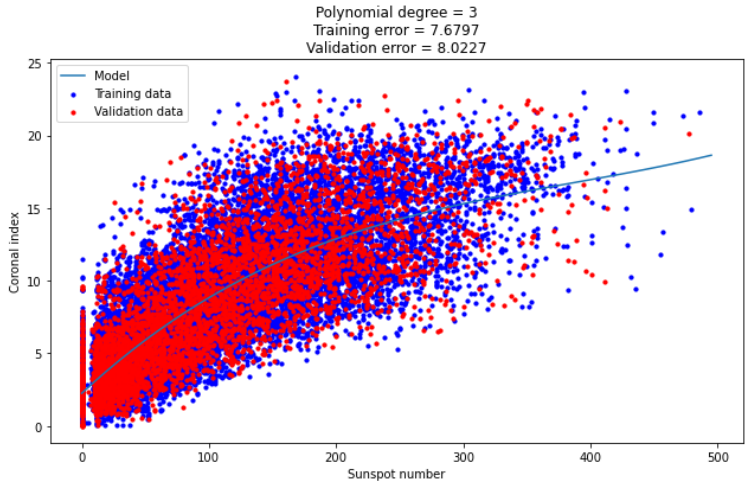


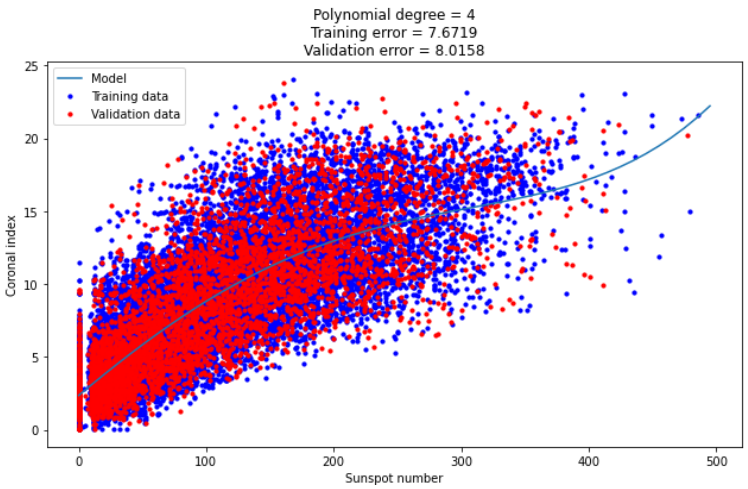
In [5]:
```
X = data['Sunspot number'].to_numpy().reshape(-1, 1)
y = data['Coronal index'].to_numpy()

X_train, X_rem, y_train, y_rem = train_test_split(X, y, test_size = 0.4, random_state = 42)
X_val, X_test, y_val, y_test = train_test_split(X_rem, y_rem, test_size = 0.5, random_state = 42)
```

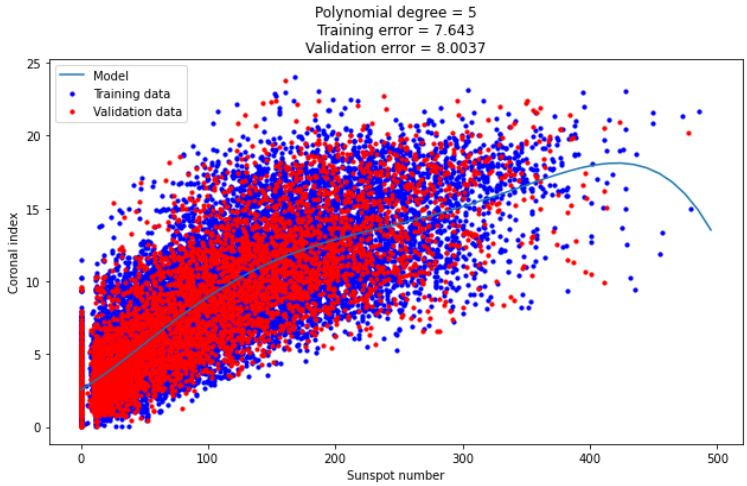
In [6]:
```
degrees = [1, 2, 3, 4, 5, 6]

tr_errors = []
val_errors = []
test_errors = []

plt.figure(figsize = (9, 40))

for i, degree in enumerate(degrees):
    plt.subplot(len(degrees), 1, i + 1)

    lin_regr = LinearRegression(fit_intercept = False)
    poly = PolynomialFeatures(degree = degree)
    X_train_poly = poly.fit_transform(X_train)
    lin_regr.fit(X_train_poly, y_train)

    y_pred_train = lin_regr.predict(X_train_poly)
    tr_error = mean_squared_error(y_train, y_pred_train)
    X_val_poly = poly.transform(X_val)
    y_pred_val = lin_regr.predict(X_val_poly)
    val_error = mean_squared_error(y_val, y_pred_val)
    X_test_poly = poly.transform(X_test)
    y_pred_test = lin_regr.predict(X_test_poly)
    test_error = mean_squared_error(y_test, y_pred_test)

    tr_errors.append(tr_error)
    val_errors.append(val_error)
    test_errors.append(test_error)

    X_fit = np.linspace(0, 495)
    plt.tight_layout()
    plt.plot(X_fit, lin_regr.predict(poly.transform(X_fit.reshape(-1, 1))), label = 'Model')
    plt.scatter(X_train, y_train, color = 'b', s = 10, label = 'Training data')
    plt.scatter(X_val, y_val, color = 'r', s = 10, label = 'Validation data')
    plt.xlabel('Sunspot number', size = 10)
    plt.ylabel('Coronal index', size = 10)
    plt.legend(loc = 'best')
    plt.title(f'Polynomial degree = {degree}\nTraining error = {tr_error:.5}\nValidation error = {val_error:.5}', size = 12)

plt.show()
```

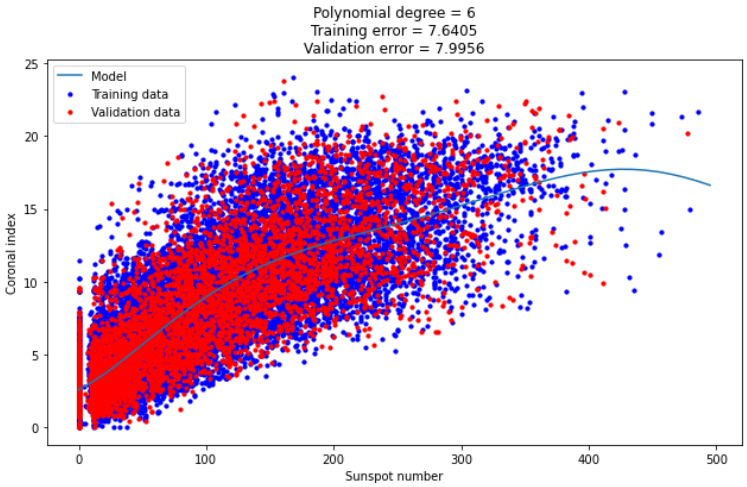



Polynomial degree = 2
Training error = 7.7033
Validation error = 8.0425
Model
Training data
Validation data
Coronal index
Sunspot number


Polynomial degree = 3
Training error = 7.6797
Validation error = 8.0227
Model
Training data
Validation data
Coronal index
Sunspot number


Polynomial degree = 4
Training error = 7.6719
Validation error = 8.0158
Model
Training data
Validation data
Coronal index
Sunspot number



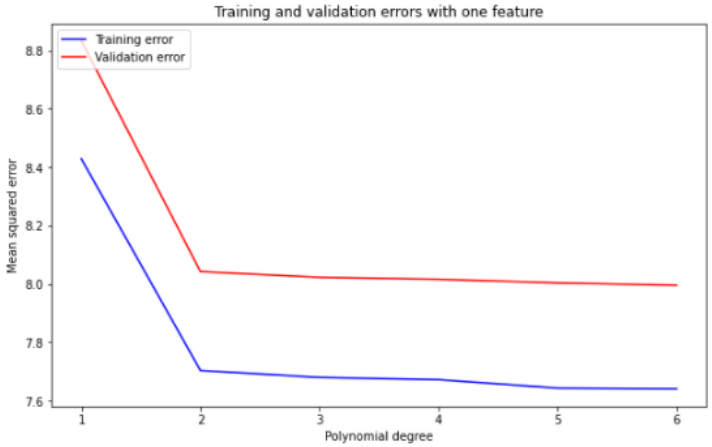


In [7]:
```
errors = {'Polynomial degree': degrees, 'Training error': tr_errors, 'Validation error': val_errors, 'Test error': test_errors}

pd.DataFrame({key: pd.Series(value) for key, value in errors.items()})
```

Out[7]:

	Polynomial degree	Training error	Validation error	Test error
0	1	8.428827	8.833335	8.671963
1	2	7.703289	8.042481	7.966550
2	3	7.679684	8.022726	7.942861
3	4	7.671935	8.015845	7.943895
4	5	7.642963	8.003734	7.936372
5	6	7.640522	7.995622	7.914492

In [8]:
```
plt.figure(figsize = (10, 6))

plt.plot(degrees, tr_errors, color = 'b', label = 'Training error')
plt.plot(degrees, val_errors, color = 'r', label = 'Validation error')
plt.legend(loc = 'upper left')
plt.xlabel('Polynomial degree', size = 10)
plt.ylabel('Mean squared error', size = 10)
plt.title('Training and validation errors with one feature', size = 12)

plt.show()
```



In [9]:
```
data['Pre 1'] = data['Sunspot number'].shift(1)
data['Pre 2'] = data['Sunspot number'].shift(2)
data['Pre 3'] = data['Sunspot number'].shift(3)
data['Pre 4'] = data['Sunspot number'].shift(4)

data = data.iloc[4:]
```

In [10]:
```
X = np.transpose(np.array([data['Sunspot number'], data['Pre 1'], data['Pre 2'], data['Pre 3'], data['Pre 4']]))
y = data['Coronal index'].to_numpy()

X_train, X_rem, y_train, y_rem = train_test_split(X, y, test_size = 0.4, random_state = 42)
X_val, X_test, y_val, y_test = train_test_split(X_rem, y_rem, test_size = 0.5, random_state = 42)
```

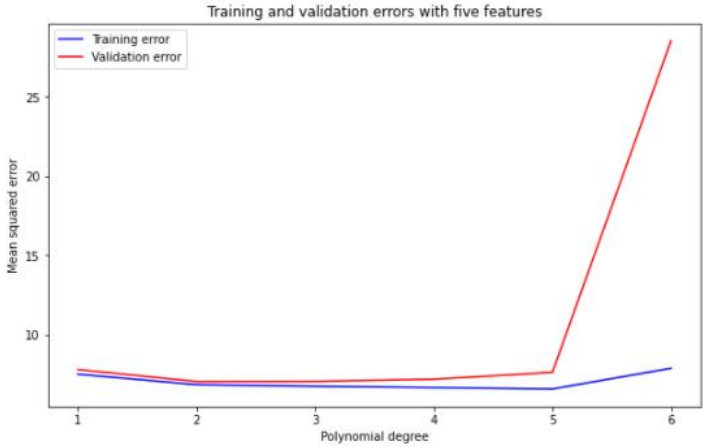
In [11]:
```
degrees = [1, 2, 3, 4, 5, 6]

tr_errors = []
val_errors = []
test_errors = []

for i, degree in enumerate(degrees):
    lin_regr = LinearRegression(fit_intercept = False)
    poly = PolynomialFeatures(degree = degree)
    X_train_poly = poly.fit_transform(X_train)
    lin_regr.fit(X_train_poly, y_train)
    y_pred_train = lin_regr.predict(X_train_poly)
    tr_error = mean_squared_error(y_train, y_pred_train)
    X_val_poly = poly.transform(X_val)
    y_pred_val = lin_regr.predict(X_val_poly)
    val_error = mean_squared_error(y_val, y_pred_val)
    X_test_poly = poly.transform(X_test)
    y_pred_test = lin_regr.predict(X_test_poly)
    test_error = mean_squared_error(y_test, y_pred_test)

    tr_errors.append(tr_error)
    val_errors.append(val_error)
    test_errors.append(test_error)
```

In [12]:
```
errors = {'Polynomial degree': degrees, 'Training error': tr_errors, 'Validation error': val_errors, 'Test error': test_errors}

pd.DataFrame({key: pd.Series(value) for key, value in errors.items()})
```

Out[12]:

	Polynomial degree	Training error	Validation error	Test error
0	1	7.515352	7.804029	7.578720
1	2	6.852728	7.053289	7.073610
2	3	6.754261	7.058372	7.040180
3	4	6.670343	7.198424	7.016341
4	5	6.580968	7.630091	7.795033
5	6	7.888221	28.558834	12.206873

In [13]:
```
plt.figure(figsize = (10, 6))

plt.plot(degrees, tr_errors, color = 'b', label = 'Training error')
plt.plot(degrees, val_errors, color = 'r', label = 'Validation error')
plt.legend(loc = 'upper left')
plt.xlabel('Polynomial degree', size = 10)
plt.ylabel('Mean squared error', size = 10)
plt.title('Training and validation errors with five features', size = 12)

plt.show()
```


Training and validation errors with five features
Training error
Validation error
Mean squared error
Polynomial degree
25
20
15
10
1
2
3
4
5
6