AI for the Industry 4.0

Artur Biedermann

Benefits of automatic verification of sensor production

- reduction of production implementation time
- increasing competitiveness on the market
- implementation of industry 4.0

Collection of measurements

Data set contains 10 sensors . Each sensor was measured three times that you can find in three separated files(air, water and isopropanol). Each file contains two-dimensional signal (signal wavelength, signal amplitude)

```
data
```

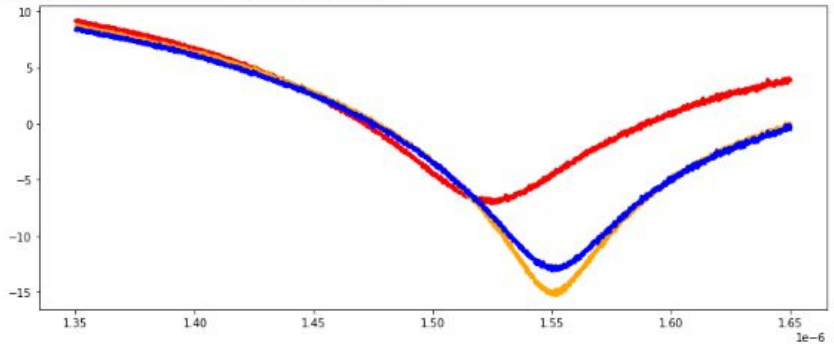
	Wavelength_air	Amplitude_air	Wavelength_izo	Amplitude_izo	Wavelength_water	Amplitude_water
0	0.000001	9.1792	0.000001	8.55510	0.000001	8.35700
1	0.000001	9.2003	0.000001	8.67500	0.000001	8.42870
2	0.000001	9.1968	0.000001	8.71160	0.000001	8.44220
3	0.000001	9.1028	0.000001	8.71570	0.000001	8.41440
4	0.000001	9.0601	0.000001	8.67920	0.000001	8.40340
...	...	...	...	...	...	...
30005	0.000002	3.2784	0.000002	-0.61478	0.000002	-0.86565
30006	0.000002	3.4213	0.000002	-0.53629	0.000002	-0.84435
30007	0.000002	3.3112	0.000002	-0.81838	0.000002	-0.97383
30008	0.000002	3.4500	0.000002	-0.57202	0.000002	-0.63849
30009	0.000002	3.6370	0.000002	-0.42871	0.000002	-0.75215

30010 rows × 6 columns

Example characteristics of one sensor

```
[7] #Plot contains one sensor in air, water and iso
    plt.rcParams["figure.figsize"] = [10.50, 4.50]
    plt.rcParams["figure.autolayout"] = True
    plt.subplot(111)
    plt.plot(sensor01_air.Wavelength_air, sensor01_air.Amplitude_air, color='red', lw=3)
    plt.plot(sensor01_izo.Wavelength_izo, sensor01_izo.Amplitude_izo, color='orange', lw=3)
    plt.plot(sensor01_water.Wavelength_water, sensor01_water.Amplitude_water, color='blue', lw=3)
```

```
[<matplotlib.lines.Line2D at 0x7f19a407b250>]
```



Building some regression models to predict values

```
#Polynomail Regression
```

```
[ ] from sklearn.preprocessing import PolynomialFeatures

    poly_reg = PolynomialFeatures(degree=2)

    X_train_2_d = poly_reg.fit_transform(x_train)
    X_test_2_d = poly_reg.transform(x_test)

    lin_reg = LinearRegression(normalize=True)
    lin_reg.fit(X_train_2_d,y_train)

    test_pred = lin_reg.predict(X_test_2_d)
    train_pred = lin_reg.predict(X_train_2_d)

    print('Test set evaluation:\n_____________________________________')
    print_evaluate(y_test, test_pred)
    print('====================================')
    print('Train set evaluation:\n_____________________________________')
    print_evaluate(y_train, train_pred)

    results_df_2 = pd.DataFrame(data=[["Polynomail Regression", *evaluate(y_test, test_pred), 0]],
                                columns=['Model', 'MAE', 'MSE', 'RMSE', 'R2 Square', 'Cross Validation'])
    results_df = results_df.append(results_df_2, ignore_index=True)
```

```
Test set evaluation:
_____________________________________
MAE: 0.6136173719953388
MSE: 0.9961778824080987
RMSE: 0.9980871116330973
R2 Square 0.9792779215487343
_____________________________________
====================================
```

Models comparison

Predicing amplitude for sensor in water

	Model	MAE	MSE	RMSE	R2 Square	Cross Validation
0	Linear Regression	2.046714	6.569168	2.563039	0.863351	0.863663
1	Ridge Regression	2.731020	10.766127	3.281178	0.776048	0.646372
2	Polynomail Regression	0.613617	0.996178	0.998087	0.979278	0.000000
3	Lasso Regression	2.735471	10.771320	3.281969	0.775939	0.757002
4	Elastic Net Regression	2.735463	10.770766	3.281884	0.775951	0.725316
5	SVM Regressor	1.024764	4.701571	2.168311	0.902200	0.000000

Predicing amplitude for sensor in isopropanol

	Model	MAE	MSE	RMSE	R2 Square	Cross Validation
0	Linear Regression	1.758077	4.743169	2.177882	0.888726	0.889047
1	Ridge Regression	2.560460	8.982286	2.997046	0.789276	0.644383
2	Polynomail Regression	0.493670	0.613493	0.783258	0.985608	0.000000
3	Lasso Regression	2.566751	8.987569	2.997927	0.789152	0.769125
4	Elastic Net Regression	2.566740	8.987030	2.997838	0.789165	0.733529
5	SVM Regressor	0.868985	3.932670	1.983096	0.907740	0.000000

In both cases Polynomail Regression have the best results