Analysis of Industrial Sensor Data Using Statistical and Regression Methods

Katalin FERENCZ
Doctoral School of Applied Informatics and Applied Mathematics - Óbuda University, Budapest, Hungary
Faculty of Technical and Human Sciences, Târgu Mureș, Romania - Sapientia Hungarian University of Transylvania
ferenczkatalin@stud.uni-obuda.hu
ferenczkatalin@ms.sapientia.ro

József DOMOKOS
Faculty of Technical and Human Sciences, Târgu Mureș
Sapientia Hungarian University of Transylvania
Târgu Mureș, Romania
domi@ms.sapientia.ro

Levente KOVÁCS
John Von Neumann Faculty of Informatics
Óbuda University
Budapest, Hungary
kovacs@uni-obuda.hu

***Abstract*— Today's industrial landscape is primarily driven by rapid and effective data processing and evaluation. Consequently, industries should devote considerable attention and resources towards real-time examination of the large data sets acquired, enabling timely extraction of vital information for outlier detection, fake data identification, and predictive analysis to mitigate unforeseen expenses. This rigorous process of data analysis necessitates the employment of a diverse set of algorithms that align with the specific objectives, spanning a wide spectrum of potential solutions. In this manuscript, we demonstrate how Apache Spark's unified engine can be harnessed for conducting statistical analysis of time series data, thereby expediting industrial data analysis processes. Furthermore, we examine and implement both linear and random forest regression models within the context of the demonstrated use case.**

Keywords*— *IoT, IIoT, regression models, algorithms, predictions, outlier detections, Apache Spark

I. INTRODUCTION

The rapid development of the industry and IT (Information Technology) and the constant technological renewal provide more and more opportunities for companies operating in the industrial sector. The Internet of Things (IoT) is becoming more widespread in industry than the Industrial Internet of Things (IIoT). As a result, the integration of various sensors and data sources increases exponentially the amount of data that can be extracted, which usually traditional algorithms are not able to process optimally in real time. Thus, there is a need to use algorithms that also use intelligent machine learning and artificial intelligence. This conducted us to expand our data processing systems with capabilities that provide valuable information based on historical and real-time data to improve current performance, make estimates, and detect underlying trends and correlations.

Industry, especially the manufacturing industry, is a data-rich environment where monitoring of assets and processes continuously generates data that is only partially used by many manufacturers. Manufacturers operate high-cost local or cloud infrastructure to store this large and ever-expanding amount of data. Intelligent data analysis as well as the use of real-time monitoring systems can provide a solution to create value from the various data stored. An important goal of the ongoing 4th Industrial Revolution, or Industry 4.0, is to create value, generate relevant knowledge, and even turn it into a business advantage for manufacturers, by interpreting the data collected. Predictive maintenance of industrial equipment has become a defining aspect of Industry 4.0 and is a critical area today. As a result of conventional reactive maintenance, where system repairs are performed after the fault has been detected, the whole process shut down is compulsory and the repair costs are high. Based on the acquired and analyzed data, predictive maintenance is possible, thus enabling the prevention of equipment failure and thus cost savings.

Due to the pace of development of industry and technology, the continuous monitoring and maintenance of systems and machines has become important, and the information obtained from them allows for fault diagnosis. Based on this, preventive or predictive strategies can be developed and applied, thus reducing the maintenance costs of machines in the long run and increasing the service life of the equipment [1]. At the same time, unplanned production downtime can be avoided. In order to achieve the right result or extract useful information from the collected data, calculations must be performed, algorithms must be run, decision-making mechanisms must be applied, and the decision made must be executed with the best possible response time and limited resources. However, it is also important to keep in mind that the time and resources available may change during the operation of the system.

Today, the industry is already trying to use various diagnostic and monitoring tools that supervises industrial processes, detect faults, deviations from a defined normal mode of operation, and in these cases different strategies are developed to signal or manage different situations. To use the appropriate diagnostic system, a faultless system model and a management system model describing typical faults are created and then used for fault diagnosis [2]. A large amount of data is collected and stored during the monitoring of the state of the system and the supervision of its processes. The analysis and interpretation of this data facilitates the above-mentioned processes. Among the simplest methods is the preparation of statistical analyses, which provide valuable status information beyond the knowledge of the system actuators. Furthermore, the application of various models (such as regression models) on the data set can provide very valuable information for predictions and decision-making.

Cite as: K. Ferencz, J. Domokos, and L. Kovács, "Analysis of Industrial Sensor Data Using Statistical and Regression Methods", *Syst. Theor. Control Comput. J.*, vol. 3, no. 1, pp. 36–44, Jun. 2023.
DOI: 10.52846/stccj.2023.3.1.48

This manuscript expands upon our preliminary findings, initially presented at the 2022 IEEE 16th International Symposium on Applied Computational Intelligence and Informatics (SACI), "*A statistical approach to time series sensor data evaluation using Apache Spark modules*" [3]. Herein, we provide a more comprehensive methodology, augmented data set, and an extended review and analysis, thereby offering a more in-depth understanding of the subject matter.

II. Real-Time Systems and Regression Models

Each of the physical processes used in the industry takes place at a given time, tied to time constraints that determine the activity and nature of the entire system. Thus, time appears as a constraint in the interpretation of systems. These systems, which take time into account as a key factor, are called real-time systems. Real-time systems maintain constant contact with their environment, thus obtaining information about the state of the environment. The information is collected with the help of sensors, and then it is transformed into useful knowledge by algorithms, and based on different decision-making mechanisms, various intervening actuators and executors perform actions in order to achieve the desired state and result [4][5][6].

There are several decision-making mechanisms described in the literature [5]:

- reactive agents: the execution of a basic action depending on the state of the outside world, based on a predetermined strategy;
- thinking agents: calculation, analysis, decision are needed to select the right action; planning agents: makes an estimate of the future state of the environment and the impact of actions.

Regression models find their greatest utility in contexts where the dependent variable under study is continuous, encompassing a multitude of potential values. Such instances might include temperature measurements, among other continuous data types. The primary objective of using regression models in these contexts is to determine the degree of influence of the explanatory variable against the dependent variable [7].

These models afford the capacity to manage numerous variables concurrently, thereby facilitating exploration of the interrelationships among these variables. Furthermore, regression models demonstrate substantial efficacy in the analysis of time series data. They also furnish the means to generate predictions upon the extant data [8].

There are several regression models that cover different application areas, the most commonly used regression models are:

- Linear Regression: it can be used to model the linear relationships between the dependent variable and the independent variable(s), the effect of each variable can be clearly illustrated from the model [9];
- Polynomial Regression: can be used if there is a nonlinear relationship between the independent variable and the dependent variable;
- Logistic Regression: used for binary target-dependent variables, models the logarithm of the odds as a function of the independent variables;
- Support Vector Regression: it is based on the basic principle of the support vector machine and effectively handles non-linear relationships and high-dimensional data;
- Random Forest Regression: able to handle non-linear relationships and complex interactions between independent variables, for which it combines many decision trees to build the final model [10].

As part of our research, we explored the implementation of both linear and random forest regression models in more detail, and we have interpreted their results during our use case. Also, we show that regression models are suitable for real time systems.

III. Industrial Application Of Intelligent Algorithms

In the intelligent manufacturing environment, the installation of various intelligent sensor and data acquisition units can collect large amounts of data (event and status information) about the operation of the system and the nature of the processes. Building on this data, it will be possible to use several frameworks in the industry, especially in the manufacturing environment, that allow for conscious statistical data analysis and evaluation based on real-time and historical data, as well as for dynamic response and real-time optimization of production. These analyzes provide predictive data that can be used to create self-adjusting mechanisms, dynamically changeable models, and alarm systems to help keep the production system running normally, or to support the rapid resumption of a critical outage, or even prevent it from occurring. An example of such a system is the IDARTS (Intelligent Data Analysis and Real-Time Supervision) framework presented by Ricardo Silva Peres in his study [11]. The Intelligent Predictive Maintenance (IPdM) System [12] presented by K. Wang also presents an industry-based system for processing real-time information and assisting in diagnostics and prognosis based on it. The distributed framework presented by Zhiqiang Ge [13] examines plant-wide industrial processes and then forecasts and diagnoses key performance indices. In his study Dazhong Wu [14] presents a fog calculation system that uses cloud-based machine learning algorithms that play an important role in process observation and predictive analysis during manufacturing processes.

In this article [15], L. Magadán presents a low-cost, self-implemented real-time monitoring system that meets the requirements of Industry 4.0. The article presents a prototype of IIoT that uses low-cost sensors and gateways, combining various analyzes and fog calculations to support the implementation of predictive maintenance. The data required for implementation were stored in the cloud and a predictive model based on machine learning was used to achieve the desired result.

In our previously published papers presenting an earlier phase of our research [16], we have introduced the architecture of our data collection and processing system. This paper focuses on an expansion of the data processing component of our system, which we have recently developed and will be presented herein.

IV. Application of Apache Spark

Nowadays, the industry is most concerned with how it can meet the expectations set for Industry 4.0, keep pace with the rapid development of technology, and achieve cost-effective day-to-day operations. To do this, however, industry must also adapt to new technologies and take advantage of the opportunities these offer.

In the following, we will present our system that also helps in industrial fast data analysis and detection of outliers in time-series of sensor values.

We are working on a prototype system that will allow easy storage of data collected from a large number of different types of smart devices in the Apache Cassandra open-source distributed NoSQL database management system [17]. Building on this, Apache Spark, also using an open-source data processing system, will allow the user to take advantage of Spark's ability to perform instant data analysis on demand. Apache Spark provides several options for data analysis:

- Spark Streaming (Near Real-Time) – scalable and fault-tolerant stream processing system;
- SparkSQL (Structured Data) – a module to support working with structured data;
- MLlib (Machine Learning) – scalable machine learning library;
- GraphX (Graph Analysis) – module specializing in graphs and graph-parallel calculations.

The Apache Spark and Cassandra architecture (Fig. 1) clearly shows the connections and communication options between the different modules.

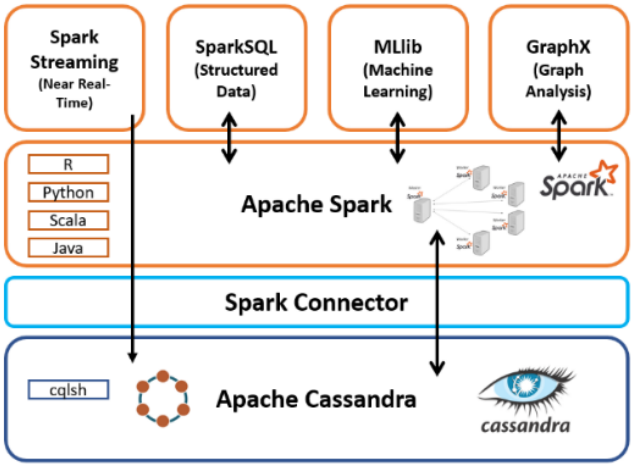


Fig. 1. Apache Spark and Cassandra Architecture

Our prototype uses Apache Cassandra 3.11.11 and Apache Spark 3.2.1 versions. The connection between the two was realized with the help of spark-cassandra-connector-assembly-3.1.0, and we chose Python from the several programming languages provided by Spark (R, Python, Scala, Java). Within Spark, the SparkSQL and MLlib modules have been integrated so far, so we present some key statistical data analysis and processing options using them.

Apache Spark is a widely used and supported open-source tool for statistical analysis, machine learning and data science. The purpose of using Spark MLlib libraries is to provide a high-level and easy-to-use set of APIs for performing machine-learning-type and various data analysis tasks.

For our implementation we used the data sets of the open-source factory system called Combined Cycle Power Plant (CCPP) [17] describing 6 years of operation, which has 9568 data points, and all data points has 5 sensor values (ambient temperature, ambient pressure, relative humidity, vacuum and electric power). This data was inserted into the database using a CQL (Cassandra Query Language) command and can then be read or modified using the assembly connector between Spark-Cassandra. As already mentioned, we chose the Python programming language, so we use the PySpark interface for data access. Depending on the data processing needs, we can change how many slaves we create for the spark master, that is how many resources (CPU, memory, disk, etc.) we allocate to the Spark application to implement data processing as quickly as possible. With this in mind, we can create our SparSession in the PySpark and then the SQLContext to query the data in the Cassandra database.

```
spark = SparkSession.builder.
appName('SparkCassandraTest').config('spark.serializer',
'org.apache.spark.serializer.KryoSerializer').config('spark.cassandr
a.connection.host', localhost).getOrCreate()

sqlContext = SQLContext(sparkContext=spark.sparkContext,
sparkSession=spark)
```

The data needed to be processed is stored in a DataFrame, which is a distributed collection for organized data in Spark.

```
df_data = sqlContext.read.
format('org.apache.spark.sql.cassandra').options(table='olddata2',
keyspace='spark_cass_data').load().select("AT", "AP", "RH", "PE",
"V").toDF("temperature", "pressure", "humidity", " electric power
", "vacuum")
```

Below we take a look at some of the very important functionalities of Spark that allow us to easily and quickly retrieve valuable information from stored data, analyze our data, detect outliers, and make predictions using a linear regression model.

With Spark's built-in functions, we can calculate some descriptive statistics in a matter of seconds. The following descriptive statistics are examined: mean, standard deviation, minimum, maximum, median, variance, skewness, kurtosis, covariance, and correlation.

1) Mean, Standard deviation, minimum and maximum values: these statistics can be calculated for the selected columns in a few seconds using the *summary()* command:

a) count – number of items in that column;

b) mean – arithmetic mean (sample mean), undistorted estimate of expected value;

c) stddev – standard deviation, the mean squared deviation from the arithmetic mean;

d) min. and max. – determination of minimum and maximum value;

e) 25%, 50%, 75% – approximate percentage of quartiles;

```
df_data.summary().show()
```

TABLE I. SUMMARY OF ALL SENSOR DATA

summary	temperature	pressure	humidity	electric power	vacuum
count	9568	9568	9568	9568	9568
mean	19.651	1013.259	73.308	454.365	54.305
stddev	7.452	5.938	14.600	17.066	12.707
min	1.81	992.89	25.56	420.26	25.36
25%	13.51	1009.1	63.32	439.75	41.74
50%	20.34	1012.94	74.96	451.51	52.08
75%	25.72	1017.26	84.83	468.43	66.54
max	37.11	1033.3	100.16	495.76	81.56

2) Median – the mean value of the data set, gives a robust estimate and is not sensitive to extreme values.
The median is calculated using the {(n+1)/2} formula, where n is the ordered row of sensor values.

df_data.agg(func.percentile_approx("temperature",0.5).alias("median")).show()

TABLE II. THE MEDIAN VALUE OF ALL SENSOR DATA

median				
temperature	*pressure*	*humidity*	*electric power*	*vacuum*
20.34	1012.94	74.96	451.51	52.08

3) Variance: it is used to describe the fluctuation of the data around the mean, that is, the value of the variance is small when our data move around the mean. The variance corresponds to the square of the standard deviation. (Standard deviation - how much our values deviate from the average.) This is calculated according to the following formula:

$$S^2 = \frac{\sum(x_i-\bar{x})^2}{n-1} \quad (1)$$

where S^2 is the sample variance, x_i is the value of the one observation, $\bar{x}$ is the mean value of all observations and n is the number of observations.
df_data.agg({'temperature': ' variance'}).show()

TABLE III. THE VARIANCE VALUE OF ALL SENSOR DATA

variance				
temperature	*pressure*	*humidity*	*electric power*	*vacuum*
55.539	35.269	213.167	291.282	161.490

4) Skewness: determines the offset of the peak of the distribution from the center position. This is calculated according to the following formula:

$$skewness = \frac{1}{n}\frac{\sum_{j=1}^{n}(x_j-\bar{x})^3}{s^3} \quad (2)$$

If the value is positive then the skew of the distribution is to the right - the mean shifts upward, the mean is greater than the median. If the value is negative, then the skew of the distribution is to the left - the mean shifts downward, there are small outliers, the mean is less than the median. If the value is zero, then the data are normally distributed, the data series is not skewed in either direction.
df_data.agg({'temperature': 'skewness'}).show()

TABLE IV. THE SKEWNESS VALUE OF ALL SENSOR DATA

skewness				
temperature	*pressure*	*humidity*	*electric power*	*vacuum*
-0.136	0.265	-0.431	0.306	0.198

5) Kurtosis: an indicator that describes the shape of the data set vertically. This is calculated according to the following formula:

$$kurtosis = \frac{1}{n}\frac{\sum_{i=1}^{n}(x_i-\bar{x})^4}{s^4} \quad (3)$$

If the value is positive, then the distribution is more peaky than normal distribution. If the value is negative, then the distribution is flatter than normal. If the value is zero, then the distribution is normal.
df_data.agg({'temperature': 'kurtosis'}).show()

TABLE V. THE KURTOSIS VALUE OF ALL SENSOR DATA

kurtosis				
temperature	*pressure*	*humidity*	*electric power*	*vacuum*
-1.037	0.093	-0.444	-1.048	-1.444

6) Histogram: with this representation, it is possible to display frequency distributions.
df_data.hist(bins = 50, figsize = (10,7.5))

Based on the obtained results (Fig. 2), the temperature (AT) and exhaust vacuum (V) show a somewhat bimodal distribution. Furthermore, the ambient pressure (AP) follows a normal distribution, while the relative humidity exhibits a left skewness.

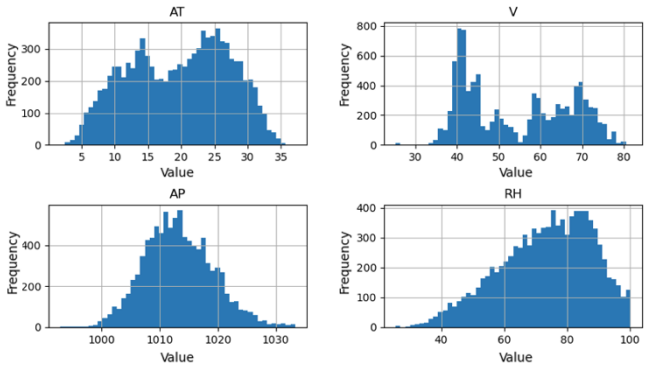


Fig. 2. Histogram of AT, AP, RH and V

7) Covariance: gives the co-movement of two different variables. This is calculated according to the following formula:

$$cov(x,y) = \frac{\sum_{i=1}^{n}(x_i-\bar{x})\,(y_i-\bar{y})}{n-1} \quad (4)$$

If the value is positive, then the relationship between X and Y is positive and if X is large, then Y is also large, and if X is small, then Y is also small. If the value is negative, then the relationship between X and Y is negative, and if X is large, then Y is small, and if X is small, then Y is large. If the value is close to zero, then there is no linear relationship between the two variables X and Y.
Based on the relationships described in Tükefci's article [18], which describes that the value of electrical power depends on changes in temperature, humidity, pressure, and vacuum, we calculated the covariance between the electrical power (X) and the other 4 sensors (Y).
df_data.cov('temperature','electric power')

TABLE VI. COVARIANCE VALUES

covariance – electric power			
temperature	*pressure*	*humidity*	*vacuum*
-120.593	52.546	97.129	-188.642

8) Correlatation: gives the magnitude and direction of the linear relationship between two values, that is it defines

their relationship to each other. This is calculated according to the following formula:

$$cor(x,y) = \frac{cov(x,y)}{S_x S_y} \quad (5)$$

The minimum value is -1 and the maximum value is 1. If the value is 1, then the relationship between the two variables is perfect for direct proportionality. If the value is 0, then there is no relationship between the two variables, so they are independent. If the value is -1, then the two variables are perfectly consistent, but the nature of the relationship is inversely proportional.

Based on the relationships described in Tükefci's article [19], which describes that the value of electrical power depends on changes in temperature, humidity, pressure, and vacuum, we calculated the correlation between the electrical power (X) and the other 4 sensors (Y).

df_data.corr('temperature','electric power')

TABLE VII. CORRELATION VALUES

correlation – electric power			
temperature	***pressure***	***humidity***	***vacuum***
-0.948	0.518	0.389	-0.869

The electrical power seems to be highly correlated with the temperature and vacuum and shows inverse proportionality.

If we plot the obtained correlation values (Fig. 3) with the help of HeatMap, it can be concluded that there is a high correlation between the temperature and the electrical power, as well as a high correlation between the electrical power and the exhaust vacuum. Furthermore, AP and RH show a positive correlation with EP, while AT and V show a strong negative correlation.

corr = df_data.corr()
sns.heatmap(corr,annot=True,cmap='YlGnBu')

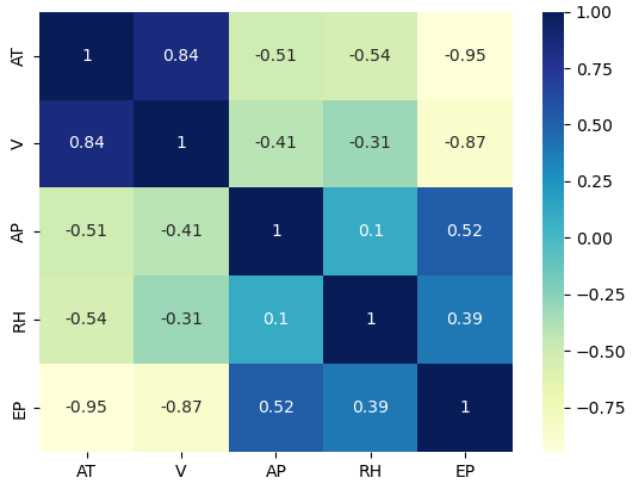


Fig. 3. Correlation HeatMap

The insight provided by the correlation helps in establishing the model of the system.

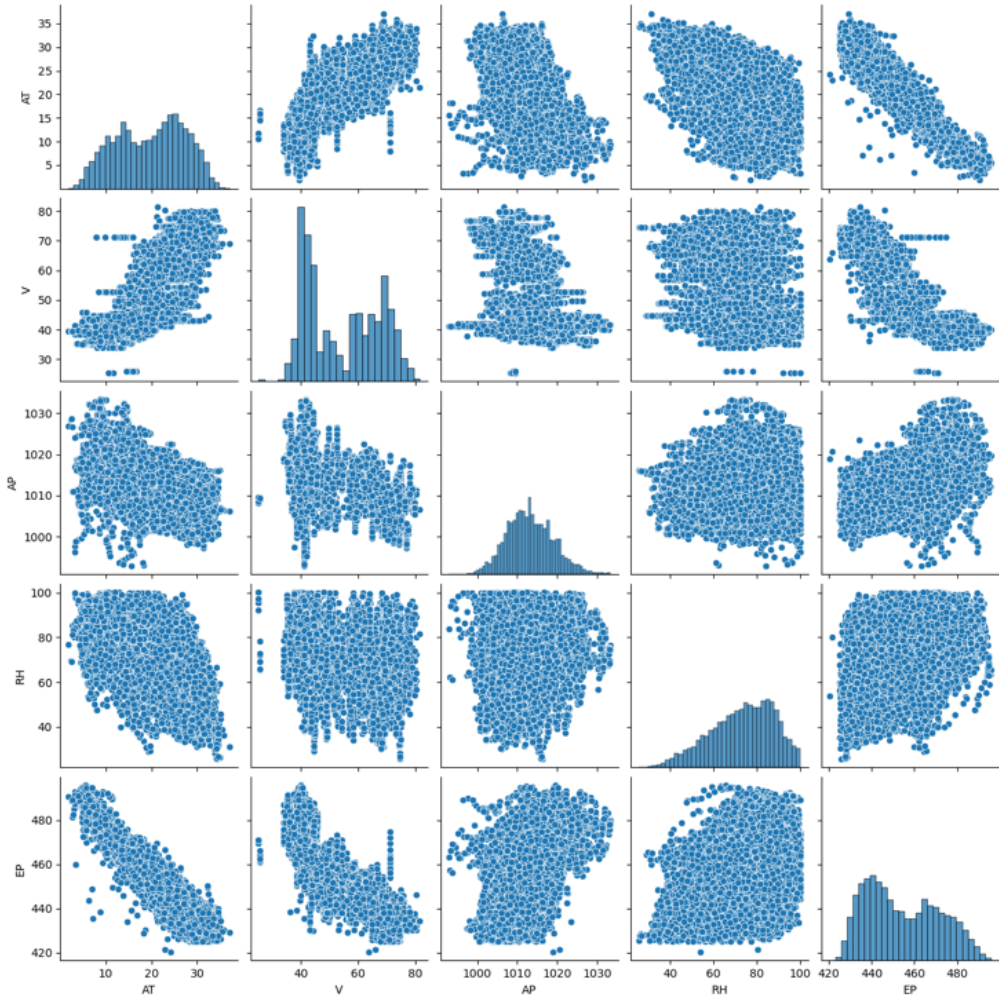


Fig. 4. The result of the parameters displayed by the pairplot function

With the *pairplot* function, the bivariate distributions in the data set for several pairs can be represented and analyzed. When using the *pairplot* function (Fig. 4), the following can be determined about our data set:

- strong positive correlation between AT and V (0.84): this is called multicollinearity, which is generally not good for systems, because our functions should be independent of each other;
- strong negative correlation between AT and EP (-0.95): if the ambient temperature increases, the electrical energy output decreases;
- strong negative correlation between V and EP (-0.87): if the exhaust vacuum is high, the emitted energy is very low.

```
sns.pairplot(df_data)
```

With Apache Spark, we can also detect outliers by running some functions and commands. During the test, we will look at the values of all sensors. The task will be to find all abnormal measurements from the DataFrame. To do this, we need to calculate the upper and lower threshold values, which are usually 3 standard deviations away from the mean of the distribution.

$$outlier_{thresholds} = mean \pm 3 * stddev \quad (6)$$

Measurements above the upper limit or below the lower limit will be considered as outliers.

In the following, only the code created for the values of the pressure sensor will be described, but the same can be done for the other sensors.

```
df_AP = sqlContext.read.
format('org.apache.spark.sql.cassandra').options(table='olddata2',
keyspace='spark_cass_data').load().select("id","AP").toDF("id","
pressure")
```

The mean and standard deviation of the sensor data can then be determined. Then, using these, we create 2 new columns for the DataFrame for the outlier thresholds (defined by the formula indicated above).

```
df_AP_mean = df_AP.agg({'pressure' : 'mean'})
df_AP_stddev = df_AP.agg({'pressure' : 'stddev'})
df_AP_stat = df_AP_mean.
withColumn("Stddev",lit(df_AP_stddev.collect()[0][0])).withColu
mn("UpperLimit", lit(df_AP_mean.collect()[0][0] +
(df_AP_stddev.collect()[0][0] * 3))).withColumn("LowerLimit",
lit(df_AP_mean.collect()[0][0] - (df_AP_stddev.collect()[0][0] *
3)))
```

Next, we link the DataFrame containing our sensor data to the new DataFrame. The result is a DataFrame with mean, standard deviation, and upper and lower thresholds for each sensor value.

```
joinDF = df_AP.join(df_AP_stat)
```

Based on this information, we can filter out rows that fall outside the range enclosed by the specified outlier thresholds.

```
def detect_outlier(values, UpperLimit, LowerLimit):
...     return (values < LowerLimit) or (values > UpperLimit)
udf_detect_outlier = udf(lambda pressure, UpperLimit,
LowerLimit: detect_outlier(pressure, UpperLimit, LowerLimit),
BooleanType())
outlierDF = joinDF.withColumn("isOutlier",
udf_detect_outlier(joinDF.pressure, joinDF.UpperLimit,
joinDF.LowerLimit)).filter("isOutlier")
outlierDF.count()
51
```

The results show that the detection of the outlier value found 51 pressure values outside the upper and lower outlier threshold intervals. From the results of the other sensor value tests, we found that in addition to the pressure values, the humidity values also contain outlier values. No outliers were found for the other sensor values.

If we plot these sensor data using the box plot function, we get the following diagrams, where we can see the outliers marked with black dots (Fig. 5).

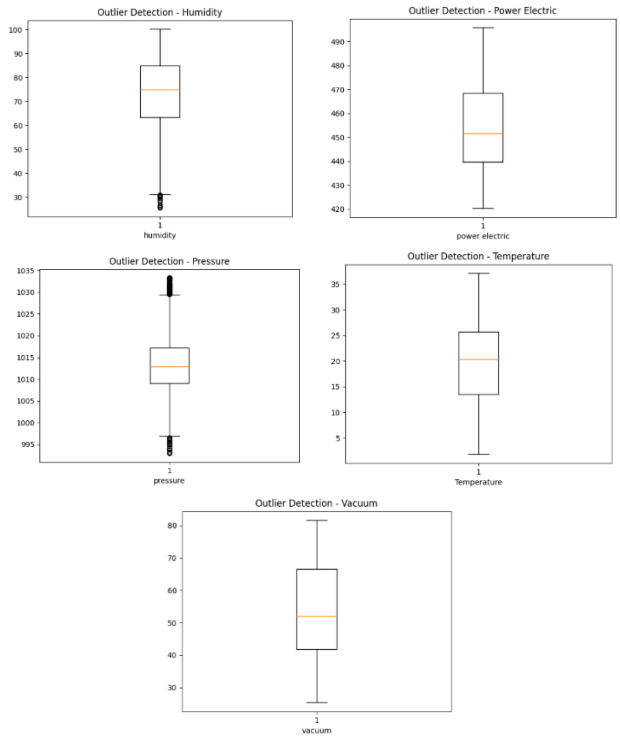


Fig. 5. Outlier Detection – box plot

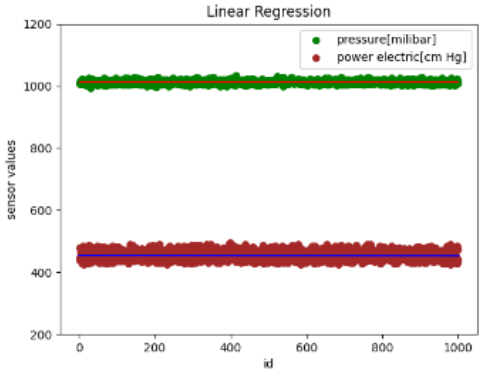
Apache Spark's MLlib libraries also allow us to make estimates. Linear regression is one of the most widely used predictive modeling methods. The purpose of the analysis is also to check whether the independent variable explains the dependent variable.

We present three solutions for regression.

A. *Matplot and Numpy*

With the use of the *matplot* and *numpy* libraries, we can easily plot our linear regression line and selected pressure values. In order to make our diagram interpretable and easy to see, we plot all 5-sensor values side by side, so we examine them together.

The resulting graph (Fig.6) clearly shows how the data points of the sensors are located relative to the linear line.



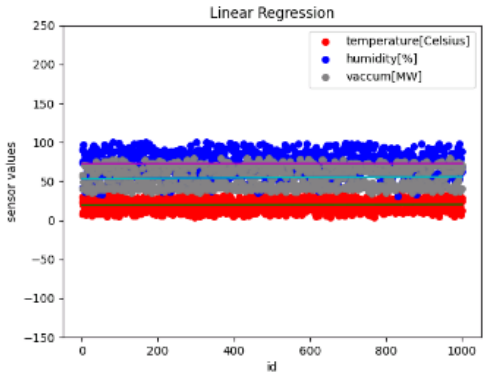


Fig. 6. Linear Regression

B. *Determination of linear regression using PySpark MLlib*

To perform linear regression calculations, we need to import three newer types of libraries into the PySpark environment:

```
from pyspark.ml.feature import VectorAssembler
from pyspark.ml.regression import LinearRegression
from pyspark.ml.evaluation import RegressionEvaluator
```

To construct a linear regression model, we use the full data set, all 5 sensor values, from which the electric power will be the label and the other four sensor values will match the feature. Features are all independent variables that we believe help predict the value of a dependent variable. The label is a dependent variable, the value of which will be predicted by our model.

VectorAssebler sums all the features into one vector, resulting in a DataFrame.

```
vectorAssembler = VectorAssembler(inputCols = ['temperature',
'pressure', 'humidity', 'vacuum'], outputCol = 'features')
v_df_data = vectorAssembler.transform(df_data)
v_df_data = v_df_data.select(['features', 'power electric'])
```

We can then divide our data into two parts: train (70%) and test (30%) data. We use the train data to teach our model based on certain algorithms and then perform the prediction on the test data.

```
splits = v_df_data.randomSplit([0.7,0.3])
train_df = splits[0]
test_df = splits[1]
```

After importing the *LinearRegression* package, we use it as an algorithm for predictive modeling to create the model. The *fit()* method is used to transmit training data for training the model, so our model predicts the values of the test data.

```
lr = LinearRegression(featuresCol = 'features', labelCol='electric
power', maxIter=10, regParam=0.3, elasticNetParam=0.8)
lr_model = lr.fit(train_df)
lr_predictions = lr_model.transform(test_df)
lr_predictions.select("prediction","electric power","features")
.show(5)
```

TABLE VIII. PREDICTION OF ELECTRIC POWER VALUES

prediction	electric power	features
489.94377352648746	490.55	[1.81, 1026.92, 76.…]
489.88982637976596	490.34	[2.34, 1028.47, 69.…]
489.565899055038	488.69	[2,58, 1028.68, 69.…]
486.27927130701426	485.2	[3.0, 1011.0, 80.14…]
482.1807303693204	489.38	[3.26, 996.32, 100.…]

We use an evaluator to check how well our model predicts the label, that is, in our case, the electric power values, and whether selecting linear regression as an algorithm for our model was a good choice. Regression model evaluation was performed using the Spark ML *RegressionEvaluator*. The metrics used are R^2 and RMSE (square root error).

```
lr_evaluator =
RegressionEvaluator(predictionCol="prediction",labelCol="electri
c power",metricName="r2")
print("R Squared (R2) on test data = %g" %
lr_evaluator.evaluate(lr_predictions))
R Squared (R2) on test data = 0.926259

test_result = lr_model.evaluate(test_df)
print("Root Mean Squared Error (RMSE) on test data = %g" %
test_result.rootMeanSquaredError)
Root Mean Squared Error (RMSE) on test data = 4.75025
```

The value of R^2 depends largely on how we select the training and test data, so the result may vary based on this. In our case, R^2 approaches 1, so it fits well. This means that the sensor values selected in features explain and affect the label (electric power) values. RMSE measures the differences between the values estimated by the model and the actual values.

When we plot the predicted values obtained during the linear regression model compared to the test data set, which was separated from the original data set using the *regplot* function, we can observe the result shown in the figure below.

```
sns.regplot(x= lr_predictions, y=test_df, lowess=True,
line_kws={'color': 'red'})
```

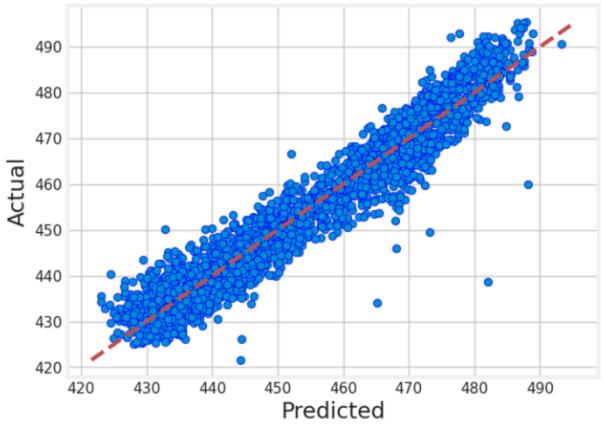


Fig. 7. Representation of the predicted and test datasets

It can be observed that the predicted values of the electric power are mostly located near to the line.

By drawing the regression diagram (Fig. 8), it is possible to examine what trends can be observed for the monthly or annual analysis, since the data set and the accompanying documentation do not contain information about time stamps, so the location of the data points within the 6-year time axis cannot be precisely determined.

```
sns.regplot(x=df_data.index, y=df.EP, x_bins=72)
sns.regplot(x=df_data.index, y=df.EP, x_bins=6)
```

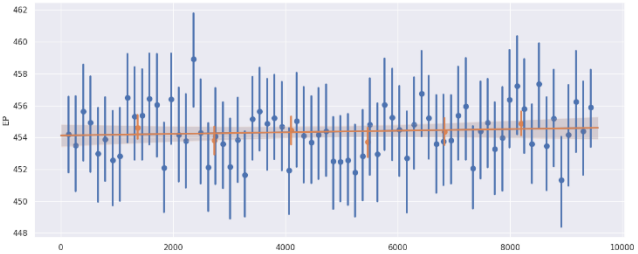


Fig. 8. Regression diagram

Testing for stationarity is frequently used in autoregressive modeling. By applying an extended Dicky-Fuller test to the electrical power data of the data set, its stationarity can be examined. This means that, an insight into the time dependence of the data can be obtained.

ADF Statistic: -66.527046
p-value: 0.000000
Critical Values:
1%: -3.431
5%: -2.862
10%: -2.567

From the obtained result, it can be concluded that the data show stationarity, since the value of the p-value is lower than the significance level (0.05), so it has no trend and shows a constant deviation over time.

C. Random Forest Regression

Similar to linear regression, we also applied the Random Forest Regression technique to the data set, so it is possible to perform both regression and classification tasks, that is, several decision trees are combined to make predictions. The importance of the model comes from its ability to measure and rank the importance of input characteristics in predicting the target variable.

```
rfr = RandomForestRegressor(n_estimators = 100, random_state = 0)
rfr.fit(X_train, y_train)
```

By plotting the test and predicted values together (Fig. 9), we can see that the two sets of values show a very high level of coverage.

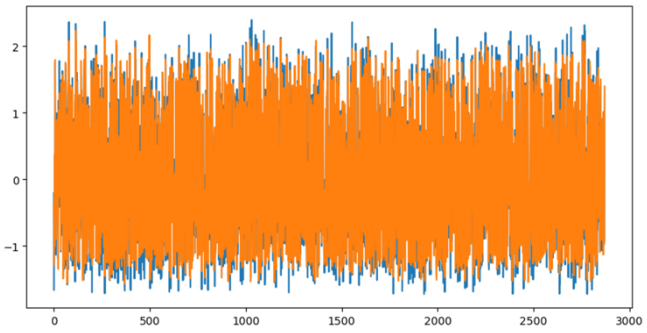


Fig. 9. Representation of predicted values and the test data set as a result of Random Forest Regression (orange: predicted, blue: actual)

From the obtained results, it can be seen that our Random Forest Regression model shows better performance than linear regression. The R2 value of 0.956 means that 95.6% of the variation of the EP target variable can be explained by the model, and the RMSE also gives a smaller value.

RMSE: 0.209
R2: 0.956

Therefore, it can be concluded that the use of Random Forest Regression can be more effective compared to linear regression.

V. Conclusions

It is important to keep in mind that in the industry, the processing and analysis of data and the extraction of useful information by using some method or algorithm is an important aspect. To this end, improvements and innovations in data analysis and processing need to be continuously developed and implemented.

In this paper we have developed and presented an alternative solution that allows us to obtain the most important descriptive statistical parameters in a simple way, as well as an alternative in the field of outlier detection and prediction. For this, we used both Apache Cassandra and Apache Spark open-source shared database and data processing systems, which are playing an increasingly important role in the intelligent, machine learning algorithms used for real-time data analysis.

As a result of the statistical models used, we showed that the analysis is consistent with the operating principles and conclusions described in Tüfekci's article [19].

The descriptive statistics and the graphical representations of sensor data are in consent with the normal operation of the combined cycle power plant.

In the scope of this study, we conducted an evaluation of the efficacity of linear and random forest models, as subsets of regression methodologies, in generating more appropriate and precise outcomes within the area of predictive analytics.

We have demonstrated how Apache Spark and Cassandra Architecture can be usefully used for time series analysis and we created an environment in which other statistical analysis can be performed.

As an enhancement, the goal is to integrate the Apache Spark GraphX and Spark Streaming modules into our system and create a unified user interface that allows to get the metrics, errors, and predictions we want with a few clicks and configurations. Also, we want to extend the used data processing algorithms and try different time series data.

Acknowledgment

This work was supported by the Collegium Talentum Programme of Hungary.

References

[1] Várkonyi-Kóczy, A.R., J.Z. Szabó. "Soft Computing Based Methods in Diagnostics" (Lágyszámítási módszerekkel támogatott diagnosztikai módszerek), XXI. Nemzetközi Gépészeti Találkozó, OGÉT'2013, (V-J. Csibi ed., 460 p.,, EMT, Cluj, Romania), Arad, Romania, Apr. 25-28, 2013, pp. 427-430.

[2] Várkonyi-Kóczy, Annamária R., Péter Baranyi, and Ron J. Patton. "Anytime fuzzy modeling approach for fault detection systems." Proceedings of the 20th IEEE Instrumentation Technology Conference (Cat. No. 03CH37412). Vol. 2. IEEE, 2003.

[3] Ferencz, Katalin, József Domokos, and Levente KovÁcs. "A statistical approach to time series sensor data evaluation using Apache Spark modules." 2022 IEEE 16th International Symposium on Applied Computational Intelligence and Informatics (SACI). IEEE, 2022.

[4] Samu, Gabor, and A. R. Várkonyi-Kóczy. "Intelligent monitor for anytime systems." IEEE International Symposium on Intelligent Signal Processing, 2003. IEEE, 2003.

[5] Samu, Gabor. Intelligent monitor for anytime systems. Diss. 2005.

[6] Khan, Md Saikat Islam, et al. "IoT and Wireless Sensor Networking-based Effluent Treatment Plant Monitoring System." Acta Polytechnica Hungarica 18.10 (2021).

[7] Forkuor, Gerald, et al. "High resolution mapping of soil properties using remote sensing variables in south-western Burkina Faso: a comparison of machine learning and multiple linear regression models." PloS one 12.1 (2017): e0170478.

[8] Ali, Iftikhar, et al. "Review of machine learning approaches for biomass and soil moisture retrievals from remote sensing data." Remote Sensing 7.12 (2015): 16398-16421.

[9] Montgomery, Douglas C., Elizabeth A. Peck, and G. Geoffrey Vining. *Introduction to linear regression analysis*. John Wiley & Sons, 2021.

[10] Coulston, John W., et al. "Approximating prediction uncertainty for random forest regression models." Photogrammetric Engineering & Remote Sensing 82.3 (2016): 189-197.

[11] Peres, Ricardo Silva, et al. "IDARTS–Towards intelligent data analysis and real-time supervision for industry 4.0." Computers in industry 101 (2018): 138-146.

[12] Wang, K.. "Intelligent Predictive Maintenance (IPdM) System – Industry 4.0 Scenario." WIT transactions on engineering sciences 113 (2016): 259-268.

[13] Ge, Zhiqiang. "Distributed predictive modeling framework for prediction and diagnosis of key performance index in plant-wide processes." Journal of Process Control 65 (2018): 107-117.

[14] Wu, Dazhong, et al. "A fog computing-based framework for process monitoring and prognosis in cyber-manufacturing." Journal of Manufacturing Systems 43 (2017): 25-34.

[15] Magadán, L., et al. "Low-cost real-time monitoring of electric motors for the Industry 4.0." Procedia Manufacturing 42 (2020): 393-398.

[16] Katalin, Ferencz and József, Domokos. "Ipari IoT szolgáltatások és nyílt forráskódú rendszerek áttekintése: Overview of Industrial IoT services and open source systems." Energetika-Elektrotechnika–Számítástechnika és Oktatás Multi-konferencia (2020): 69-74.

[17] Ferencz, Katalin, and József Domokos. "Rapid Prototyping of IoT Applications for the Industry." 2020 IEEE International Conference on Automation, Quality and Testing, Robotics (AQTR). IEEE, 2020.

[18] https://archive.ics.uci.edu/ml/datasets/combined+cycle+power+plant

[19] Pınar Tüfekci, Prediction of full load electrical power output of a base load operated combined cycle power plant using machine learning methods, International Journal of Electrical Power & Energy Systems, Volume 60, September 2014, pp. 126-140