ARTIFICIAL INTELLIGENCE FINAL REPORT

VSB Power Line Fault Detection Classification

22.06.2021

Ayşe AKIŞIK, Betül Berna SOYLU

171805007, 171805019

1.) CHOOSE DATASET

➔ We choose https://www.kaggle.com/c/vsb-power-line-fault-detection dataset.

File Descriptions

metadata_[train/test].csv

id_measurement: The ID code for a trio of signals recorded at the same time.

signal_id: The foreign key for the signal data. Each signal ID is unique across both train and test, so the first ID in train is '0' but the first ID in test is '8712'.

phase: The phase ID code within the signal trio. The phases may or may not all be impacted by a fault on the line.

target: 0 if the power line is undamaged, 1 if there is a fault.

2.) PREPROCESS DATA AND EXTRACT FEATURE SET

PREPROCESSİNG OF DATA:

Some corrections made on the data set, completing the missing data, removing duplicate data, transforming, integrating, cleaning, normalizing, dimension reduction, etc. are transactions.

```
# ******************** Preprocessing ************************************************************
preprocessing_data = pd.DataFrame(StandardScaler().fit(df).transform(df))
# ***********************************************************************************************
```

EXTRACT FEATURE SET:

We labeled it 0 if the power line is undamaged and 1 if it is faulty.

```
# not fault train
notfaulttrain = extractFeature(preprocessing_data,250,10,"0")
# fault train
faulttrain = extractFeature(preprocessing_data,250,10,"1")

merged_train = pd.concat([notfaulttrain, faulttrain])

# not fault test
notfaulttest = extractFeature(preprocessing_data,250,10,"0")
# fault test
faulttest = extractFeature(preprocessing_data,250,10,"1")

merged_test = pd.concat([notfaulttest, faulttest])
```

faulttrain - DataFrame

Index	mean	std	max	min	kurtosis	zerocross	label
0	-1.68215	0.028581	-1.63284	-1.73145	-1.20004	0	1
1	-1.67817	0.028581	-1.62887	-1.72748	-1.20004	0	1
2	-1.6742	0.028581	-1.62489	-1.7235	-1.20004	0	1
3	-1.67022	0.028581	-1.62091	-1.71953	-1.20004	0	1
4	-1.66624	0.028581	-1.61694	-1.71555	-1.20004	0	1
5	-1.66227	0.028581	-1.61296	-1.71157	-1.20004	0	1
6	-1.65829	0.028581	-1.60899	-1.7076	-1.20004	0	1
7	-1.65432	0.028581	-1.60501	-1.70362	-1.20004	0	1
8	-1.65034	0.028581	-1.60103	-1.69964	-1.20004	0	1
9	-1.64636	0.028581	-1.59706	-1.69567	-1.20004	0	1
10	-1.64239	0.028581	-1.59308	-1.69169	-1.20004	0	1
11	-1.63841	0.028581	-1.5891	-1.68772	-1.20004	0	1

notfaulttrain - DataFrame

Index	mean	std	max	min	kurtosis	zerocross	label
0	-1.68215	0.028581	-1.63284	-1.73145	-1.20004	0	0
1	-1.67817	0.028581	-1.62887	-1.72748	-1.20004	0	0
2	-1.6742	0.028581	-1.62489	-1.7235	-1.20004	0	0
3	-1.67022	0.028581	-1.62091	-1.71953	-1.20004	0	0
4	-1.66624	0.028581	-1.61694	-1.71555	-1.20004	0	0
5	-1.66227	0.028581	-1.61296	-1.71157	-1.20004	0	0
6	-1.65829	0.028581	-1.60899	-1.7076	-1.20004	0	0
7	-1.65432	0.028581	-1.60501	-1.70362	-1.20004	0	0
8	-1.65034	0.028581	-1.60103	-1.69964	-1.20004	0	0
9	-1.64636	0.028581	-1.59706	-1.69567	-1.20004	0	0
10	-1.64239	0.028581	-1.59308	-1.69169	-1.20004	0	0
11	-1.63841	0.028581	-1.5891	-1.68772	-1.20004	0	0

3.) CLASSIFICATION ALGORITHMS

- **DECISION TREE:** Classification is a classification method that creates a model in the form of a tree structure consisting of decision nodes and leaf nodes according to feature and target. Its purpose is to create a model that predicts the value of a variable by extracting simple rules from data features and learning these rules.

```
Decision Tree Algorithm accuracy score 1.0
Confusion Matrix of Decision Tree Algorithm
[[1626   27]
 [  28   62]]
```

- **K NEAREST NEIGHBOURS (KNN):** It is an easy-to-apply supervised learning algorithm. This algorithm consists of five steps.

  1-) First of all, the K value is determined.

  2-) Euclidean distances from other objects to the target object are calculated.

  3-) The distances are listed and the nearest neighbors are found depending on the minimum distance.

  4-) Nearest neighbor categories are added.

  5-) The most suitable neighbor category is selected**.**

```
KNN Algorithm accuracy score 0.9711579853637538
Confusion Matrix of KNN Algorithm
[[1605   48]
 [  85    5]]
```

- **LOGISTIC REGRESSION:** It is a regression method for classification. It is used to classify categorical or numerical data. It works if the dependent variable, namely the result, can only take 2 different values. (Yes / No, Male / Female etc.)

```
Logistic Regression Algorithm accuracy score 0.9375807145931985
Confusion Matrix of Logistic Regression Algorithm
[[1653    0]
 [  90    0]]
```

- **NAIVE BAYES:** This classification is also based on Bayes' theorem. lazy is a learning algorithm, it can also work on unstable datasets. The way the algorithm works is it calculates the probability of each state for an element and classifies it according to the highest probability value.

```
Naive Bayes Algorithm accuracy score 0.9375807145931985
Confusion Matrix of Naive Bayes Algorithm
[[1653    0]
 [  90    0]]
```

- **RANDOM FOREST:** Random Forests algorithm, which is an ensemble learning method, is an algorithm that aims to increase the classification value by producing more than one decision tree during the classification process.

```
Random Forest Algorithm accuracy score 0.9820634237336777
Confusion Matrix of Random Forest Algorithm
[[1564   89]
 [  89    1]]
```

- **ADABOOST:** It can be used in conjunction with many other types of learning algorithms to improve performance. The output of the other learning algorithms is combined into a weighted sum that represents the final output of the boosted classifier. AdaBoost is adaptive in the sense that subsequent weak learners are tweaked in favor of those instances misclassified by previous classifiers.

```
AdaBoost Algorithm accuracy score 0.9381546850337208
Confusion Matrix of AdaBoost Algorithm
[[1652    1]
 [  90    0]]
```

- **SUPPORT VECTOR MACHİNE (SV CLASSIFIER):** It is one of the supervised learning methods used in classification problems. Draws a line to separate points placed on a plane. The purpose of this line is that it is the maximum distance for the points of both classes.

```
SVC Algorithm accuracy score 0.9375807145931985
Confusion Matrix of SVC Algorithm
[[1653    0]
 [  90    0]]
```

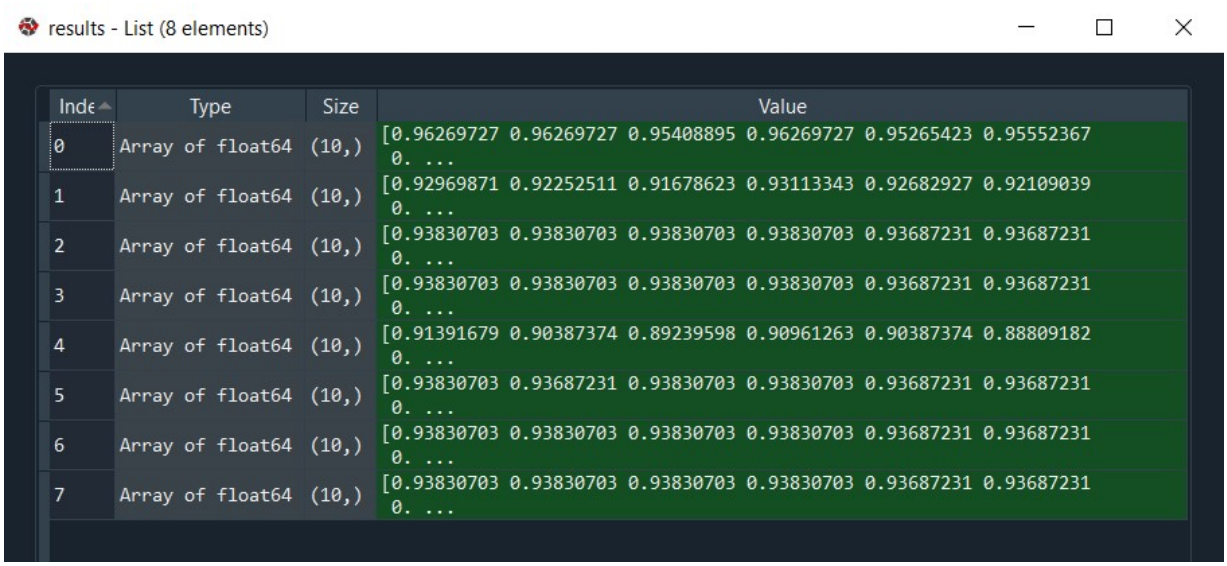
- **MULTI-LAYER PERCEPTRON (MLP/ANN):** It is a supervised learning algorithm that learns a function by training on a dataset with the number of dimensions for the input and the number of dimensions for the output. Given a set of features and a target, it can learn a nonlinear function estimator for classification.

```
MLP Algorithm accuracy score 0.9375807145931985
Confusion Matrix of MLP Algorithm
[[1653    0]
 [  90    0]]
```

4-) K-FOLD & BOXPLOTS

- We used 10-fold cross validation to compare machine learning algorithms and plot. We created an results array and append all score value of algorithms.

```
In [13]: results = []
    ...:
    ...: for model in models:
    ...:     score = cross_val_score(model, X_train, y_train, cv=10)
    ...:     results.append(score)
```



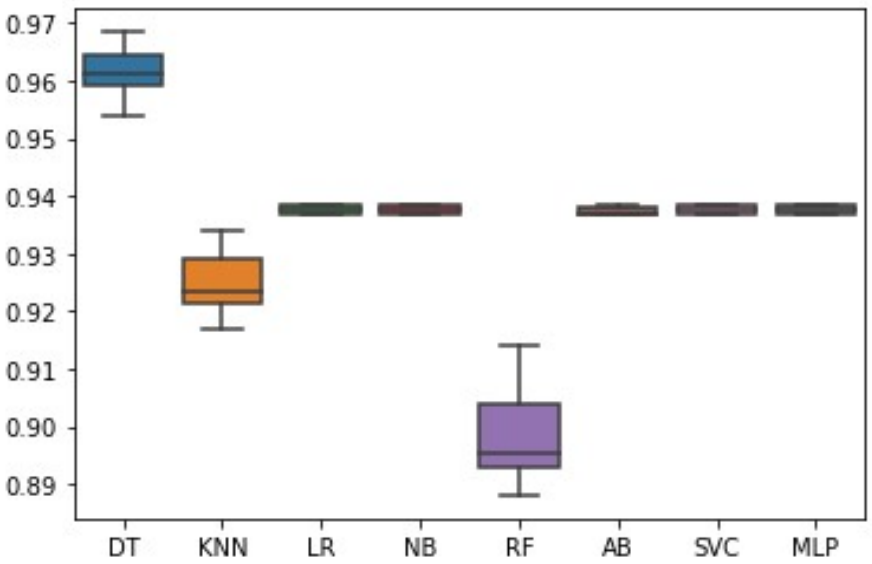
- We found the mean of 10 values for each algorithm with the "mean" function. Then we found which algorithm have the highest accuracy.

```
In [10]: puan = []
    ...:
    ...: for i in range(len(names)):
    ...:     puan.append(results[i].mean())
    ...: print("Highest accuracy value:")
    ...: print(names[puan.index(max(puan))], max(puan))
Highest accuracy value:
DT 0.9614004600999356
```

puan - List (8 elements)

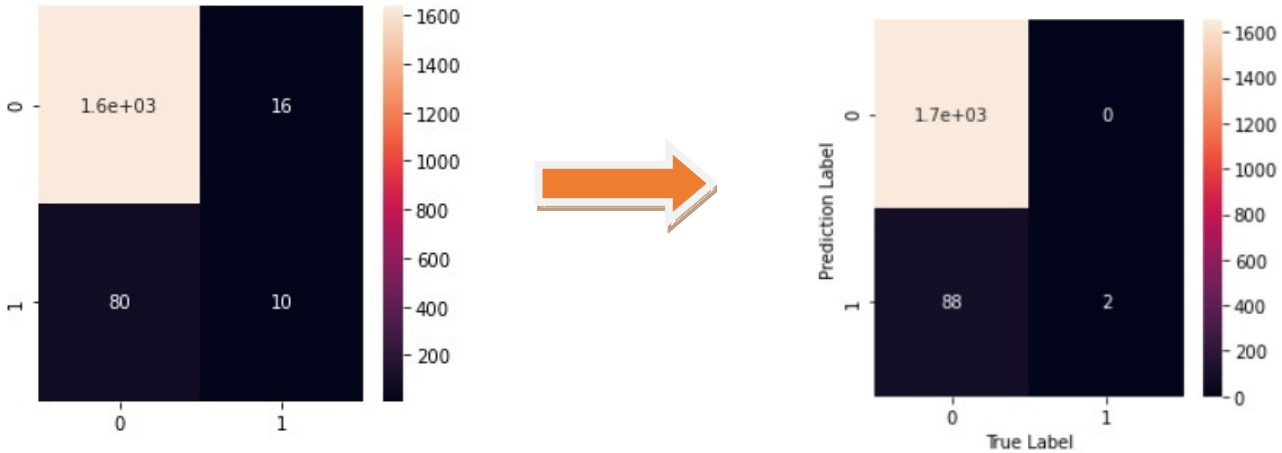
Inde	Type	Size	Value
0	float64	1	0.9614004600999356
1	float64	1	0.9245225844753376
2	float64	1	0.9375808060818945
3	float64	1	0.9375808060818945
4	float64	1	0.8988371757449827
5	float64	1	0.9372938620359836
6	float64	1	0.9375808060818945
7	float64	1	0.9375808060818945

Boxplot of Algorithms



5-) HYPERPARAMETERS

- The histogram of the entered hypermaters is shown below.



- If we compare the hyperparameter values we tried:

  ➔ While our dataset correctly calculates "1.6e+03" grain 0 values, by changing the parameters, we see that "1.7e+03" 0 values are calculated correctly. We see that the incorrectly calculated value of 0 is 0.

  ➔ In the same way, while our data set was calculating 80 0 values correctly, we trained our data by increasing the number of correct to 88 and reducing the number of errors to 2 with different parameters we tried.

6-) COMPARISON HISTOGRAM

- We created a "sorted HP" list. We use the "operator.itemgetter(1) " library to list the values and output them.
- We sort and print the "dictHP" list parameters values in the loop.
- We created a custom label grade with the plot library and then plotted its histogram.

```
import operator

sortedHP  = {}
sortedHP = sorted(hyperparameters.items(),key = operator.itemgetter(1),reverse=True)
print(sortedHP)

dictHP = {}
for hyperparameters in sortedHP[0:3]:
    dictHP[hyperparameters[0]] = hyperparameters[1]
    print(hyperparameters[1])

plt.xsticks(rotation=90)
plt.hist(dictHP)
```

7-) RANDOM INPUT PLOT

- We have created an array in the “prediction” expression. With this array, we added the entered values that we predicted to our list.
- We then plotted these values with a bar graph.

```
predictions = []
predictions.append(classifier_dt.predict([[12, 0]])[0])
predictions.append(classifier_dt.predict([[1, 1]])[0])
predictions.append(classifier_dt.predict([[5, 2]])[0])

#plot draw
plt.bar(np.arange(len(predictions)), predictions)
```

8-) ERROR

- When we wanted to read the train parquet file, the kernel was restarted.