phase2b_2_cores

January 11, 2023

1 TBD - etap 2b

1.0.1 Michał Kopyt, Rafał Kulus, Adrian Prorok

1.1 Dodatek - pomiary czasu trenowania modeli dla 2 wątków

1.2 Inicjalizacja sesji sparkowej, załadowanie bibliotek pyspark, SynapseML oraz pysparkling i połączenie do klastra H2O:

[1]:
```
pip install h2o_pysparkling_3.2
```

```
Collecting h2o_pysparkling_3.2
  Downloading h2o_pysparkling_3.2-3.38.0.4-1.tar.gz (162.2 MB)
                             162.2/162.2
MB 5.0 MB/s eta 0:00:0000:0100:01
  Preparing metadata (setup.py) … done
Requirement already satisfied: requests in /opt/conda/lib/python3.10/site-
packages (from h2o_pysparkling_3.2) (2.28.1)
Collecting tabulate
  Downloading tabulate-0.9.0-py3-none-any.whl (35 kB)
Collecting future
  Downloading future-0.18.2.tar.gz (829 kB)
                             829.2/829.2 kB
21.1 MB/s eta 0:00:0000:01
  Preparing metadata (setup.py) … done
Requirement already satisfied: charset-normalizer<3,>=2 in
/opt/conda/lib/python3.10/site-packages (from requests->h2o_pysparkling_3.2)
(2.1.0)
Requirement already satisfied: certifi>=2017.4.17 in
/opt/conda/lib/python3.10/site-packages (from requests->h2o_pysparkling_3.2)
(2022.6.15)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/opt/conda/lib/python3.10/site-packages (from requests->h2o_pysparkling_3.2)
(1.26.9)
Requirement already satisfied: idna<4,>=2.5 in /opt/conda/lib/python3.10/site-
packages (from requests->h2o_pysparkling_3.2) (3.3)
Building wheels for collected packages: h2o_pysparkling_3.2, future
  Building wheel for h2o_pysparkling_3.2 (setup.py) … done
```

```
  Created wheel for h2o_pysparkling_3.2:
filename=h2o_pysparkling_3.2-3.38.0.4.post1-py2.py3-none-any.whl size=162427908
sha256=9efd7904b692a6b4bafe79aa6e36ba2bef1b5fb33070be3e986449ae422f838f
  Stored in directory: /home/jovyan/.cache/pip/wheels/8d/b1/1a/48d776d100aa559b0
12748271998372ccae02a056f1362d95c
  Building wheel for future (setup.py) … done
  Created wheel for future: filename=future-0.18.2-py3-none-any.whl
size=491058
sha256=30b7a4d2eedb94feb574851566debe85dcd6aed0a6bd6e4ab74411213eb649a6
  Stored in directory: /home/jovyan/.cache/pip/wheels/22/73/06/557dc4f4ef68179b9
d763930d6eec26b88ed7c389b19588a1c
Successfully built h2o_pysparkling_3.2 future
Installing collected packages: tabulate, future, h2o_pysparkling_3.2
Successfully installed future-0.18.2 h2o_pysparkling_3.2-3.38.0.4.post1
tabulate-0.9.0
Note: you may need to restart the kernel to use updated packages.
```

[2]:
```
seed = 20031999
```

[3]:
```
import pyspark
from pyspark.conf import SparkConf

spark = pyspark.sql.SparkSession.builder.appName("tbd2") \
    .master("local[2]") \
    .config("spark.driver.memory", '2g') \
    .config("spark.executor.cores", 2) \
    .config("spark.jars.packages", "com.microsoft.azure:synapseml_2.12:0.9.5") \
    .config("spark.jars.repositories", "https://mmlspark.azureedge.net/maven") \
    .getOrCreate()

from pysparkling import *
import h2o
hc = H2OContext.getOrCreate()
```

```
Connecting to H2O server at http://2acda88be3a7:54323 … successful.

--------------------------  ------------------------------------------
H2O_cluster_uptime:         10 secs
H2O_cluster_timezone:       Etc/UTC
H2O_data_parsing_timezone:  UTC
H2O_cluster_version:        3.38.0.4
H2O_cluster_version_age:    1 day
H2O_cluster_name:           sparkling-water-jovyan_local-1673053479270
H2O_cluster_total_nodes:    1
H2O_cluster_free_memory:    1.869 Gb
H2O_cluster_total_cores:    4
H2O_cluster_allowed_cores:  2
H2O_cluster_status:         locked, healthy
```

```
H2O_connection_url:         http://2acda88be3a7:54323
H2O_connection_proxy:       null
H2O_internal_security:      False
Python_version:             3.10.5 final
--------------------------  ----------------------------------------


Sparkling Water Context:
 * Sparkling Water Version: 3.38.0.4-1-3.2
 * H2O name: sparkling-water-jovyan_local-1673053479270
 * cluster size: 1
 * list of used nodes:
  (executorId, host, port)
  ------------------------
  (0,172.17.0.2,54321)
  ------------------------

  Open H2O Flow in browser: http://2acda88be3a7:54323 (CMD + click in Mac OSX)
```

1.3 Importy

[4]:
```
from pyspark.sql.functions import col, to_date, month, to_timestamp, hour,␣
 ↪regexp_replace
from pyspark.ml.functions import vector_to_array
from pyspark.ml import Pipeline
from pyspark.ml.feature import StringIndexer, OneHotEncoder, VectorAssembler

from timeit import default_timer as timer
import pandas as pd

from pyspark.ml.evaluation import BinaryClassificationEvaluator,␣
 ↪MulticlassClassificationEvaluator
from pyspark.ml.tuning import CrossValidator, ParamGridBuilder
from sklearn.metrics import roc_auc_score, accuracy_score

from pyspark.ml.classification import LogisticRegression, GBTClassifier
from pysparkling.ml import H2OGLM, H2OXGBoostClassifier
from sklearn.linear_model import LogisticRegression as SklearnLogisticRegression
from synapse.ml.lightgbm import LightGBMClassifier
```

1.4 Ładowanie danych

[5]:
```
user_name = 'jovyan'

# ścieżki dostępu do plików
```

```
csv_path_1e4 = 'file:////home/jovyan/work/tbd-notebooks/data/ds1-1e4.csv'
csv_path_1e5 = 'file:////home/jovyan/work/tbd-notebooks/data/ds1-1e5.csv'
csv_path_1e6 = 'file:////home/jovyan/work/tbd-notebooks/data/ds1-1e6.csv'
```

1.5 Przygotowanie funkcji do ładowania i przygotowania danych na podstawie wybranego pliku:

[6]:
```
def get_features_df(csv_path):
    df = spark.read.csv(csv_path, inferSchema=True, header="true",␣
 ↪nullValue='NA', nanValue='NA',emptyValue='NA')
    df = df.filter('Longitude is not NULL and Latitude is not NULL')
    df = df.withColumn('label', df.label.cast('integer'))

    df = df.withColumn('Date', to_date(df.Date, 'dd/MM/yyyy'))
    df = df.withColumn('Month', month(df.Date))
    df = df.withColumn('Time', to_timestamp(df.Time, 'HH:mm'))
    df = df.withColumn('Hour', hour(df.Time))

    df = df.withColumn('Light_Conditions', regexp_replace('Light_Conditions', ':
 ↪', ''))

    df = df.drop('V1', 'Accident_Index', 'Location_Easting_OSGR',␣
 ↪'Location_Northing_OSGR', 'Accident_Severity', 'Date', 'Time',␣
 ↪'Local_Authority_(District)', 'Local_Authority_(Highway)',␣
 ↪'1st_Road_Number', '2nd_Road_Number', 'LSOA_of_Accident_Location', 'Year')

    columns_for_one_hot_encoding = ['Day_of_Week', '1st_Road_Class',␣
 ↪'Road_Type', 'Junction_Control', '2nd_Road_Class',␣
 ↪'Pedestrian_Crossing-Human_Control',␣
 ↪'Pedestrian_Crossing-Physical_Facilities', 'Light_Conditions',␣
 ↪'Weather_Conditions', 'Road_Surface_Conditions',␣
 ↪'Special_Conditions_at_Site', 'Carriageway_Hazards', 'Urban_or_Rural_Area',␣
 ↪'Did_Police_Officer_Attend_Scene_of_Accident', 'Month', 'Hour']
    other_columns = ['Longitude', 'Latitude', 'Police_Force',␣
 ↪'Number_of_Vehicles', 'Number_of_Casualties', 'Speed_limit']

    stringindexer_stages = [StringIndexer(inputCol=c,␣
 ↪outputCol='stringindexed_' + c).setHandleInvalid("keep") for c in␣
 ↪columns_for_one_hot_encoding]
    onehotencoder_stages = [OneHotEncoder(inputCol='stringindexed_' + c,␣
 ↪outputCol='onehot_' + c) for c in columns_for_one_hot_encoding]

    extracted_columns = ['onehot_' + c for c in columns_for_one_hot_encoding]
    vectorassembler_stage = VectorAssembler(inputCols=extracted_columns +␣
 ↪other_columns, outputCol='features')
```

```
    pipeline_stages = stringindexer_stages + onehotencoder_stages +␣
↪[vectorassembler_stage]

    return Pipeline(stages=pipeline_stages).fit(df).transform(df).
↪select(['features', 'label'])
```

1.6 Dodatkowe funkcje do statystyk itp.

[7]:
```
class ModelTestingResults:
  def __init__(self, training_time = 0, auc = 0, accuracy = 0, confusion_matrix␣
↪= None):
    self.training_time = training_time
    self.auc = auc
    self.accuracy = accuracy
    self.confusion_matrix = confusion_matrix

class ModelTuningResults:
  def __init__(self, tuned_params = {}, auc = 0, accuracy = 0, confusion_matrix␣
↪= None):
    self.tuned_params = tuned_params
    self.auc = auc
    self.accuracy = accuracy
    self.confusion_matrix = confusion_matrix

def get_confusion_matrix(predictions_df):
    return predictions_df.select('label', 'prediction').groupBy('label',␣
↪'prediction').count().sort(col('label'), col('prediction')).toPandas()

def get_confusion_matrix_sklearn(testing_df, sklearn_pred):
    Y_testing = testing_df.select('label').toPandas().to_numpy().ravel()
    predictions_df = pd.DataFrame(data={'prediction': sklearn_pred, 'label':␣
↪Y_testing})
    return predictions_df.
↪groupby(['label','prediction'])[['label','prediction']].size().
↪reset_index(name='count').sort_values(by=['label', 'prediction']).
↪reset_index(drop=True)

def print_model_testing_results(model_testing_results, label):
    if label:
        print(f'----- {label} -----')
    print(f'Czas trenowania: {round(model_testing_results.training_time, 3)}s')
    print(f'AUC: {round(model_testing_results.auc, 3)}')
    print(f'Accuracy: {round(model_testing_results.accuracy, 3)}')
    print('Macierz pomyłek:')
    print(model_testing_results.confusion_matrix)
```

2 Przygotowanie funkcji do testowania modeli

2.1 sparkML

[8]:
```
sparkML_evaluator_auroc = BinaryClassificationEvaluator(rawPredictionCol="rawPrediction", metricName="areaUnderROC")
sparkML_evaluator_accuracy = MulticlassClassificationEvaluator(labelCol="label", predictionCol="prediction", metricName="accuracy")

def test_sparkML(base_model, training_df, testing_df):
    training_start_time = timer()
    model = base_model.fit(training_df)
    training_end_time = timer()

    prediction_df = model.transform(testing_df)

    return ModelTestingResults(
        training_time=training_end_time - training_start_time,
        auc=sparkML_evaluator_auroc.evaluate(prediction_df),
        accuracy=sparkML_evaluator_accuracy.evaluate(prediction_df),
        confusion_matrix=get_confusion_matrix(prediction_df)
    )

def test_sparkML_lr_basic_version(training_df, testing_df):
    sparkML_lr = LogisticRegression()
    return test_sparkML(sparkML_lr, training_df, testing_df)

def test_sparkML_gbt_basic_version(training_df, testing_df):
    sparkML_gbt = GBTClassifier()
    return test_sparkML(sparkML_gbt, training_df, testing_df)
```

2.2 H2O-sparklinkg-water

[9]:
```
def test_h2o(base_model, training_df, testing_df):
    training_start_time = timer()
    model = base_model.fit(training_df)
    training_end_time = timer()

    prediction_df = model.transform(testing_df)
    predicted_labels_for_testing_data = prediction_df.withColumn('prediction', prediction_df.prediction.cast('int')).select('prediction').toPandas().to_numpy().ravel()
    labels_for_testing_data = prediction_df.select('label').toPandas().to_numpy().ravel()
```

```
    probabilities_for_1 = prediction_df.withColumn('detailed_prediction',␣
 ↪col('detailed_prediction').probabilities['1']).select('detailed_prediction').
 ↪toPandas().to_numpy().ravel()

    return ModelTestingResults(
        training_time=training_end_time - training_start_time,
        auc=roc_auc_score(labels_for_testing_data, probabilities_for_1),
        accuracy=accuracy_score(labels_for_testing_data,␣
 ↪predicted_labels_for_testing_data),
        confusion_matrix=get_confusion_matrix(prediction_df)
    )

def test_h2o_lr_basic_version(training_df, testing_df):
    h2o_lr = H2OGLM(
        family="binomial",
        featuresCols=['features'],
        labelCol='label'
    )
    return test_h2o(h2o_lr, training_df, testing_df)

def test_h2o_gbt_basic_version(training_df, testing_df):
    h2o_gbt = H2OXGBoostClassifier(featuresCols=['features'], labelCol='label')
    return test_h2o(h2o_gbt, training_df, testing_df)
```

2.3 scikit-learn (do implementacji nierozproszonej)

[10]:
```
##############################################################################################
import warnings

warnings.simplefilter(action='ignore', category=pd.errors.PerformanceWarning)
##############################################################################################

def test_sklearn_lr_basic_version(training_df, testing_df):
    sklearn_lr = SklearnLogisticRegression(n_jobs = 2)

    training_start_time = timer()
    features_num = training_df.first().features.size
    X = training_df.withColumn('x', vector_to_array('features')).
 ↪select([col('x')[i] for i in range(features_num)]).toPandas()
    y = training_df.select('label').toPandas().to_numpy().ravel()
    model = sklearn_lr.fit(X, y)
    training_end_time = timer()

    X_testing = testing_df.withColumn('x', vector_to_array('features')).
 ↪select([col('x')[i] for i in range(features_num)]).toPandas()
    y_testing = testing_df.select('label').toPandas().to_numpy().ravel()
```

```
    prediction_df = model.predict(X_testing)

    return ModelTestingResults(
        training_time=training_end_time - training_start_time,
        auc=roc_auc_score(y_testing, model.predict_proba(X_testing)[:, 1]),
        accuracy=accuracy_score(y_testing, prediction_df),
        confusion_matrix=get_confusion_matrix_sklearn(testing_df, prediction_df)
    )
```

2.4 SynapseML

[11]:
```
def test_synapseML(base_model, training_df, testing_df):
    return test_sparkML(base_model, training_df, testing_df) # Ten sam kod

def test_synapseML_gbt_basic_version(training_df, testing_df):
    synapseML_gbt = LightGBMClassifier(objective="binary",␣
 ↪featuresCol="features", labelCol="label", numThreads=2)
    return test_synapseML(synapseML_gbt, training_df, testing_df)
```

3 Regresja logistyczna - pomiary czasów trenowania modeli dla różnych zbiorów danych dla 2 wątków

3.1 Zbiór danych 1e4

[12]:
```
training_df_1e4, testing_df_1e4 = get_features_df(csv_path_1e4).randomSplit([0.
 ↪8, 0.2], seed=seed)
```

[13]:
```
sparkML_lr_basic_1e4_results = test_sparkML_lr_basic_version(training_df_1e4,␣
 ↪testing_df_1e4)

label_sparkML_lr_basic_1e4_results = 'Regresja logistyczna, sparkML, zbiór 1e4,␣
 ↪podstawowe hiperparametry, 2 wątki'
print_model_testing_results(sparkML_lr_basic_1e4_results,␣
 ↪label_sparkML_lr_basic_1e4_results)
```

```
----- Regresja logistyczna, sparkML, zbiór 1e4, podstawowe hiperparametry, 2
wątki -----
Czas trenowania: 6.75s
AUC: 0.646
Accuracy: 0.836
Macierz pomyłek:
   label  prediction  count
0      0         0.0      2
1      0         1.0    326
2      1         0.0      7
3      1         1.0   1698
```

[14]:
```
h2o_lr_basic_1e4_results = test_h2o_lr_basic_version(training_df_1e4,␣
↪testing_df_1e4)

label_h2o_lr_basic_1e4_results = 'Regresja logistyczna, H2O-sparkling-water,␣
↪zbiór 1e4, podstawowe hiperparametry, 2 wątki'
print_model_testing_results(h2o_lr_basic_1e4_results,␣
↪label_h2o_lr_basic_1e4_results)
```

```
----- Regresja logistyczna, H2O-sparkling-water, zbiór 1e4, podstawowe
hiperparametry, 2 wątki -----
Czas trenowania: 3.758s
AUC: 0.646
Accuracy: 0.837
Macierz pomyłek:
   label prediction  count
0      0          0      2
1      0          1    326
2      1          0      6
3      1          1   1699
```

[15]:
```
sklearn_lr_basic_1e4_results = test_sklearn_lr_basic_version(training_df_1e4,␣
↪testing_df_1e4)

label_sklearn_lr_basic_1e4_results = 'Regresja logistyczna, scikit-learn, zbiór␣
↪1e4, podstawowe hiperparametry, 2 wątki'
print_model_testing_results(sklearn_lr_basic_1e4_results,␣
↪label_sklearn_lr_basic_1e4_results)
```

```
----- Regresja logistyczna, scikit-learn, zbiór 1e4, podstawowe hiperparametry,
2 wątki -----
Czas trenowania: 4.839s
AUC: 0.646
Accuracy: 0.839
Macierz pomyłek:
   label  prediction  count
0      0           0      3
1      0           1    325
2      1           0      3
3      1           1   1702
```

3.2 Zbiór danych 1e5

[16]:
```
training_df_1e5, testing_df_1e5 = get_features_df(csv_path_1e5).randomSplit([0.
↪8, 0.2], seed=seed)
```

[17]:
```
sparkML_lr_basic_1e5_results = test_sparkML_lr_basic_version(training_df_1e5,␣
↪testing_df_1e5)
```

```
label_sparkML_lr_basic_1e5_results = 'Regresja logistyczna, sparkML, zbiór 1e5,␣
 ↪podstawowe hiperparametry, 2 wątki'
print_model_testing_results(sparkML_lr_basic_1e5_results,␣
 ↪label_sparkML_lr_basic_1e5_results)
```

```
----- Regresja logistyczna, sparkML, zbiór 1e5, podstawowe hiperparametry, 2
wątki -----
Czas trenowania: 9.51s
AUC: 0.666
Accuracy: 0.852
Macierz pomyłek:
   label  prediction  count
0      0         0.0     20
1      0         1.0   2928
2      1         0.0     15
3      1         1.0  16890
```

[18]:
```
h2o_lr_basic_1e5_results = test_h2o_lr_basic_version(training_df_1e5,␣
 ↪testing_df_1e5)

label_h2o_lr_basic_1e5_results = 'Regresja logistyczna, H2O-sparkling-water,␣
 ↪zbiór 1e5, podstawowe hiperparametry, 2 wątki'
print_model_testing_results(h2o_lr_basic_1e5_results,␣
 ↪label_h2o_lr_basic_1e5_results)
```

```
----- Regresja logistyczna, H2O-sparkling-water, zbiór 1e5, podstawowe
hiperparametry, 2 wątki -----
Czas trenowania: 6.128s
AUC: 0.665
Accuracy: 0.852
Macierz pomyłek:
   label prediction  count
0      0          0      7
1      0          1   2941
2      1          0      5
3      1          1  16900
```

[19]:
```
sklearn_lr_basic_1e5_results = test_sklearn_lr_basic_version(training_df_1e5,␣
 ↪testing_df_1e5)

label_sklearn_lr_basic_1e5_results = 'Regresja logistyczna, scikit-learn, zbiór␣
 ↪1e5, podstawowe hiperparametry, 2 wątki'
print_model_testing_results(sklearn_lr_basic_1e5_results,␣
 ↪label_sklearn_lr_basic_1e5_results)
```

```
----- Regresja logistyczna, scikit-learn, zbiór 1e5, podstawowe hiperparametry,
```

```
2 wątki -----
Czas trenowania: 33.389s
AUC: 0.66
Accuracy: 0.852
Macierz pomyłek:
   label  prediction  count
0      0           0     19
1      0           1   2929
2      1           0     12
3      1           1  16893
```

3.3 Zbiór danych 1e6

[20]:
```
training_df_1e6, testing_df_1e6 = get_features_df(csv_path_1e6).randomSplit([0.
↪8, 0.2], seed=seed)
```

[21]:
```
sparkML_lr_basic_1e6_results = test_sparkML_lr_basic_version(training_df_1e6,␣
↪testing_df_1e6)

label_sparkML_lr_basic_1e6_results = 'Regresja logistyczna, sparkML, zbiór 1e6,␣
↪podstawowe hiperparametry, 2 wątki'
print_model_testing_results(sparkML_lr_basic_1e6_results,␣
↪label_sparkML_lr_basic_1e6_results)
```

```
----- Regresja logistyczna, sparkML, zbiór 1e6, podstawowe hiperparametry, 2
wątki -----
Czas trenowania: 34.946s
AUC: 0.665
Accuracy: 0.851
Macierz pomyłek:
   label  prediction   count
0      0         0.0     125
1      0         1.0   29772
2      1         0.0      87
3      1         1.0  170211
```

[22]:
```
h2o_lr_basic_1e6_results = test_h2o_lr_basic_version(training_df_1e6,␣
↪testing_df_1e6)

label_h2o_lr_basic_1e6_results = 'Regresja logistyczna, H2O-sparkling-water,␣
↪zbiór 1e6, podstawowe hiperparametry, 2 wątki'
print_model_testing_results(h2o_lr_basic_1e6_results,␣
↪label_h2o_lr_basic_1e6_results)
```

```
----- Regresja logistyczna, H2O-sparkling-water, zbiór 1e6, podstawowe
hiperparametry, 2 wątki -----
Czas trenowania: 38.361s
```

```
AUC: 0.665
Accuracy: 0.851
Macierz pomyłek:
   label prediction   count
0      0          0      75
1      0          1   29822
2      1          0      54
3      1          1  170244
```

[34]:
```
sklearn_lr_basic_1e6_results = test_sklearn_lr_basic_version(training_df_1e6,␣
 ↪testing_df_1e6)

label_sklearn_lr_basic_1e6_results = 'Regresja logistyczna, scikit-learn, zbiór␣
 ↪1e6, podstawowe hiperparametry, 2 wątki'
print_model_testing_results(sklearn_lr_basic_1e6_results,␣
 ↪label_sklearn_lr_basic_1e6_results)
```

```
----- Regresja logistyczna, scikit-learn, zbiór 1e6, podstawowe hiperparametry,
2 wątki -----
Czas trenowania: 306.877s
AUC: 0.659
Accuracy: 0.851
Macierz pomyłek:
   label  prediction   count
0      0           0      74
1      0           1   29823
2      1           0      53
3      1           1  170245
```

4 Gradient Boosted Trees - pomiary czasów trenowania modeli dla różnych zbiorów danych dla 2 wątków

4.1 Zbiór danych 1e4

[24]:
```
sparkML_gbt_basic_1e4_results = test_sparkML_gbt_basic_version(training_df_1e4,␣
 ↪testing_df_1e4)

label_sparkML_gbt_basic_1e4_results = 'Gradient Boosted Trees, sparkML, zbiór␣
 ↪1e4, podstawowe hiperparametry, 2 wątki'
print_model_testing_results(sparkML_gbt_basic_1e4_results,␣
 ↪label_sparkML_gbt_basic_1e4_results)
```

```
----- Gradient Boosted Trees, sparkML, zbiór 1e4, podstawowe hiperparametry, 2
wątki -----
Czas trenowania: 11.315s
AUC: 0.675
Accuracy: 0.837
```

```
Macierz pomyłek:
   label  prediction  count
0      0         0.0      3
1      0         1.0    325
2      1         0.0      7
3      1         1.0   1698
```

[25]:
```
h2o_gbt_basic_1e4_results = test_h2o_gbt_basic_version(training_df_1e4,␣
 ↪testing_df_1e4)

label_h2o_gbt_basic_1e4_results = 'Gradient Boosted Trees, H2O-sparkling-water,␣
 ↪zbiór 1e4, podstawowe hiperparametry, 2 wątki'
print_model_testing_results(h2o_gbt_basic_1e4_results,␣
 ↪label_h2o_gbt_basic_1e4_results)
```

```
----- Gradient Boosted Trees, H2O-sparkling-water, zbiór 1e4, podstawowe
hiperparametry, 2 wątki -----
Czas trenowania: 4.858s
AUC: 0.655
Accuracy: 0.815
Macierz pomyłek:
   label prediction  count
0      0          0     56
1      0          1    272
2      1          0    104
3      1          1   1601
```

[26]:
```
synapseML_gbt_basic_1e4_results =␣
 ↪test_synapseML_gbt_basic_version(training_df_1e4, testing_df_1e4)

label_synapseML_gbt_basic_1e4_results = 'Gradient Boosted Trees, SynapseML,␣
 ↪zbiór 1e4, podstawowe hiperparametry, 2 wątki'
print_model_testing_results(synapseML_gbt_basic_1e4_results,␣
 ↪label_synapseML_gbt_basic_1e4_results)
```

```
----- Gradient Boosted Trees, SynapseML, zbiór 1e4, podstawowe hiperparametry, 2
wątki -----
Czas trenowania: 1.911s
AUC: 0.676
Accuracy: 0.838
Macierz pomyłek:
   label  prediction  count
0      0         0.0      9
1      0         1.0    319
2      1         0.0     10
3      1         1.0   1695
```

4.2 Zbiór danych 1e5

[27]:
```
sparkML_gbt_basic_1e5_results = test_sparkML_gbt_basic_version(training_df_1e5,␣
 ↪testing_df_1e5)

label_sparkML_gbt_basic_1e5_results = 'Gradient Boosted Trees, sparkML, zbiór␣
 ↪1e5, podstawowe hiperparametry, 2 wątki'
print_model_testing_results(sparkML_gbt_basic_1e5_results,␣
 ↪label_sparkML_gbt_basic_1e5_results)
```

```
----- Gradient Boosted Trees, sparkML, zbiór 1e5, podstawowe hiperparametry, 2
wątki -----
Czas trenowania: 21.94s
AUC: 0.681
Accuracy: 0.851
Macierz pomyłek:
   label  prediction  count
0      0         0.0      9
1      0         1.0   2939
2      1         0.0     16
3      1         1.0  16889
```

[28]:
```
h2o_gbt_basic_1e5_results = test_h2o_gbt_basic_version(training_df_1e5,␣
 ↪testing_df_1e5)

label_h2o_gbt_basic_1e5_results = 'Gradient Boosted Trees, H2O-sparkling-water,␣
 ↪zbiór 1e5, podstawowe hiperparametry, 2 wątki'
print_model_testing_results(h2o_gbt_basic_1e5_results,␣
 ↪label_h2o_gbt_basic_1e5_results)
```

```
----- Gradient Boosted Trees, H2O-sparkling-water, zbiór 1e5, podstawowe
hiperparametry, 2 wątki -----
Czas trenowania: 11.027s
AUC: 0.682
Accuracy: 0.847
Macierz pomyłek:
   label prediction  count
0      0          0    171
1      0          1   2777
2      1          0    264
3      1          1  16641
```

[29]:
```
synapseML_gbt_basic_1e5_results =␣
 ↪test_synapseML_gbt_basic_version(training_df_1e5, testing_df_1e5)

label_synapseML_gbt_basic_1e5_results = 'Gradient Boosted Trees, SynapseML,␣
 ↪zbiór 1e5, podstawowe hiperparametry, 2 wątki'
```

```
print_model_testing_results(synapseML_gbt_basic_1e5_results,␣
 ↪label_synapseML_gbt_basic_1e5_results)
```

```
----- Gradient Boosted Trees, SynapseML, zbiór 1e5, podstawowe hiperparametry, 2
wątki -----
Czas trenowania: 5.992s
AUC: 0.69
Accuracy: 0.852
Macierz pomyłek:
   label  prediction  count
0      0         0.0     36
1      0         1.0   2912
2      1         0.0     32
3      1         1.0  16873
```

4.3 Zbiór danych 1e6

[30]:
```
sparkML_gbt_basic_1e6_results = test_sparkML_gbt_basic_version(training_df_1e6,␣
 ↪testing_df_1e6)

label_sparkML_gbt_basic_1e6_results = 'Gradient Boosted Trees, sparkML, zbiór␣
 ↪1e6, podstawowe hiperparametry, 2 wątki'
print_model_testing_results(sparkML_gbt_basic_1e6_results,␣
 ↪label_sparkML_gbt_basic_1e6_results)
```

```
----- Gradient Boosted Trees, sparkML, zbiór 1e6, podstawowe hiperparametry, 2
wątki -----
Czas trenowania: 127.415s
AUC: 0.68
Accuracy: 0.851
Macierz pomyłek:
   label  prediction   count
0      0         0.0     159
1      0         1.0   29738
2      1         0.0      92
3      1         1.0  170206
```

[31]:
```
h2o_gbt_basic_1e6_results = test_h2o_gbt_basic_version(training_df_1e6,␣
 ↪testing_df_1e6)

label_h2o_gbt_basic_1e6_results = 'Gradient Boosted Trees, H2O-sparkling-water,␣
 ↪zbiór 1e6, podstawowe hiperparametry, 2 wątki'
print_model_testing_results(h2o_gbt_basic_1e6_results,␣
 ↪label_h2o_gbt_basic_1e6_results)
```

```
----- Gradient Boosted Trees, H2O-sparkling-water, zbiór 1e6, podstawowe
hiperparametry, 2 wątki -----
```

```
Czas trenowania: 78.894s
AUC: 0.694
Accuracy: 0.851
Macierz pomyłek:
   label prediction   count
0      0          0     663
1      0          1   29234
2      1          0     569
3      1          1  169729
```

[32]:
```
synapseML_gbt_basic_1e6_results =␣
 ↪test_synapseML_gbt_basic_version(training_df_1e6, testing_df_1e6)

label_synapseML_gbt_basic_1e6_results = 'Gradient Boosted Trees, SynapseML,␣
 ↪zbiór 1e6, podstawowe hiperparametry, 2 wątki'
print_model_testing_results(synapseML_gbt_basic_1e6_results,␣
 ↪label_synapseML_gbt_basic_1e6_results)
```

```
----- Gradient Boosted Trees, SynapseML, zbiór 1e6, podstawowe hiperparametry, 2
wątki -----
Czas trenowania: 39.783s
AUC: 0.695
Accuracy: 0.851
Macierz pomyłek:
   label  prediction   count
0      0         0.0     394
1      0         1.0   29503
2      1         0.0     260
3      1         1.0  170038
```