hw2

March 31, 2022

1 Praca domowa 2

1.1 modelowanie

1.1.1 Paulina Jaszczuk

1.1.2 Import pakietów

[230]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn import preprocessing
import xgboost as xgb

from tqdm.notebook import tqdm

from math import sqrt
from sklearn.metrics import mean_squared_error as MSE
from sklearn.metrics import roc_auc_score, accuracy_score, f1_score, roc_curve,␣
 ↪r2_score
from sklearn.ensemble import RandomForestClassifier, RandomForestRegressor,␣
 ↪BaggingClassifier

# ustawia domyślną wielkość wykresów
plt.rcParams['figure.figsize'] = (12,8)
# to samo tylko dla tekstu
plt.rcParams['font.size'] = 16
```

1.1.3 Klasyfikacja

[217]:
```
aps_train = pd.read_csv("C:
 ↪\\Users\\pauli\\Downloads\\wb_hw2_data\\airline_passenger_satisfaction\\train.
 ↪csv")
aps_test = pd.read_csv("C:
 ↪\\Users\\pauli\\Downloads\\wb_hw2_data\\airline_passenger_satisfaction\\test.
 ↪csv")
```

[218]:
```
aps_train.head()
```

[218]:
```
   Unnamed: 0      id  Gender      Customer Type  Age   Type of Travel  \
0           0   70172    Male     Loyal Customer   13  Personal Travel
1           1    5047    Male  disloyal Customer   25  Business travel
2           2  110028  Female     Loyal Customer   26  Business travel
3           3   24026  Female     Loyal Customer   25  Business travel
4           4  119299    Male     Loyal Customer   61  Business travel

      Class  Flight Distance  Inflight wifi service  \
0  Eco Plus              460                      3
1  Business              235                      3
2  Business             1142                      2
3  Business              562                      2
4  Business              214                      3

   Departure/Arrival time convenient  …  Inflight entertainment  \
0                                  4  …                       5
1                                  2  …                       1
2                                  2  …                       5
3                                  5  …                       2
4                                  3  …                       3

   On-board service  Leg room service  Baggage handling  Checkin service  \
0                 4                 3                 4                4
1                 1                 5                 3                1
2                 4                 3                 4                4
3                 2                 5                 3                1
4                 3                 4                 4                3

   Inflight service  Cleanliness  Departure Delay in Minutes  \
0                 5            5                          25
1                 4            1                           1
2                 4            5                           0
3                 4            2                          11
4                 3            3                           0

   Arrival Delay in Minutes             satisfaction
0                      18.0  neutral or dissatisfied
1                       6.0  neutral or dissatisfied
2                       0.0                satisfied
3                       9.0  neutral or dissatisfied
4                       0.0                satisfied

[5 rows x 25 columns]
```

[219]:
```
aps_train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 103904 entries, 0 to 103903
Data columns (total 25 columns):
 #   Column                             Non-Null Count   Dtype
---  ------                             --------------   -----
 0   Unnamed: 0                         103904 non-null  int64
 1   id                                 103904 non-null  int64
 2   Gender                             103904 non-null  object
 3   Customer Type                      103904 non-null  object
 4   Age                                103904 non-null  int64
 5   Type of Travel                     103904 non-null  object
 6   Class                              103904 non-null  object
 7   Flight Distance                    103904 non-null  int64
 8   Inflight wifi service              103904 non-null  int64
 9   Departure/Arrival time convenient  103904 non-null  int64
 10  Ease of Online booking             103904 non-null  int64
 11  Gate location                      103904 non-null  int64
 12  Food and drink                     103904 non-null  int64
 13  Online boarding                    103904 non-null  int64
 14  Seat comfort                       103904 non-null  int64
 15  Inflight entertainment             103904 non-null  int64
 16  On-board service                   103904 non-null  int64
 17  Leg room service                   103904 non-null  int64
 18  Baggage handling                   103904 non-null  int64
 19  Checkin service                    103904 non-null  int64
 20  Inflight service                   103904 non-null  int64
 21  Cleanliness                        103904 non-null  int64
 22  Departure Delay in Minutes         103904 non-null  int64
 23  Arrival Delay in Minutes           103594 non-null  float64
 24  satisfaction                       103904 non-null  object
dtypes: float64(1), int64(19), object(5)
memory usage: 19.8+ MB
```

Jest kilka zmiennych kategorycznych.

[220]:
```
aps_train.isna().sum()
```

[220]:
```
Unnamed: 0                           0
id                                   0
Gender                               0
Customer Type                        0
Age                                  0
Type of Travel                       0
Class                                0
Flight Distance                      0
Inflight wifi service                0
Departure/Arrival time convenient    0
Ease of Online booking               0
```

```
Gate location                        0
Food and drink                       0
Online boarding                      0
Seat comfort                         0
Inflight entertainment               0
On-board service                     0
Leg room service                     0
Baggage handling                     0
Checkin service                      0
Inflight service                     0
Cleanliness                          0
Departure Delay in Minutes           0
Arrival Delay in Minutes           310
satisfaction                         0
dtype: int64
```

Braki danych w kolumnie `Arrival Delay in Minutes`

Kodowanie zmiennych kategorycznych

[221]:
```
# Zmienne 'Gender', 'Customer Type', 'Tyoe of Travel', 'satisfaction' mają po 2␣
↪wartości - kodujemy je binarnie
aps_train['Gender'] = (aps_train['Gender'] == 'Female')*1
aps_train['Customer Type'] = (aps_train['Customer Type'] == 'Loyal Customer')*1
aps_train['Type of Travel'] = (aps_train['Type of Travel'] == 'Personal␣
↪Travel')*1
aps_train['satisfaction'] = (aps_train['satisfaction'] == 'satisfied')*1

# 'Class' przyjmuje 3 wartości - robimy one hot encoding i wyrzucamy jedną␣
↪kolumnę, żeby uniknąć liniowości
encoded_train = pd.get_dummies(aps_train[["Class"]].astype(str))
encoded_train = encoded_train.drop(["Class_Eco Plus"], axis = 1)
aps_train = pd.concat([aps_train, encoded_train], axis = 1)

aps_train = aps_train.drop(columns=["Unnamed: 0","Class"],axis=1)#dropping


aps_test['Gender'] = (aps_test['Gender'] == 'Female')*1
aps_test['Customer Type'] = (aps_test['Customer Type'] == 'Loyal Customer')*1
aps_test['Type of Travel'] = (aps_test['Type of Travel'] == 'Personal Travel')*1
aps_test['satisfaction'] = (aps_test['satisfaction'] == 'satisfied')*1

encoded_test = pd.get_dummies(aps_test[["Class"]].astype(str))
encoded_test = encoded_test.drop(["Class_Eco Plus"], axis = 1)
aps_test = pd.concat([aps_test, encoded_test], axis = 1)

aps_test = aps_test.drop(columns=["Unnamed: 0","Class"],axis=1)#dropping
```

Braki danych

[222]:
```
# Usuwamy wiersze z brakami danych
aps_train.drop(aps_train[aps_train["Arrival Delay in Minutes"].isnull()].index,
 ↪axis=0, inplace=True)
aps_test.drop(aps_test[aps_test["Arrival Delay in Minutes"].isnull()].index,
 ↪axis=0, inplace=True)
```

Podział zbioru na train, val i test

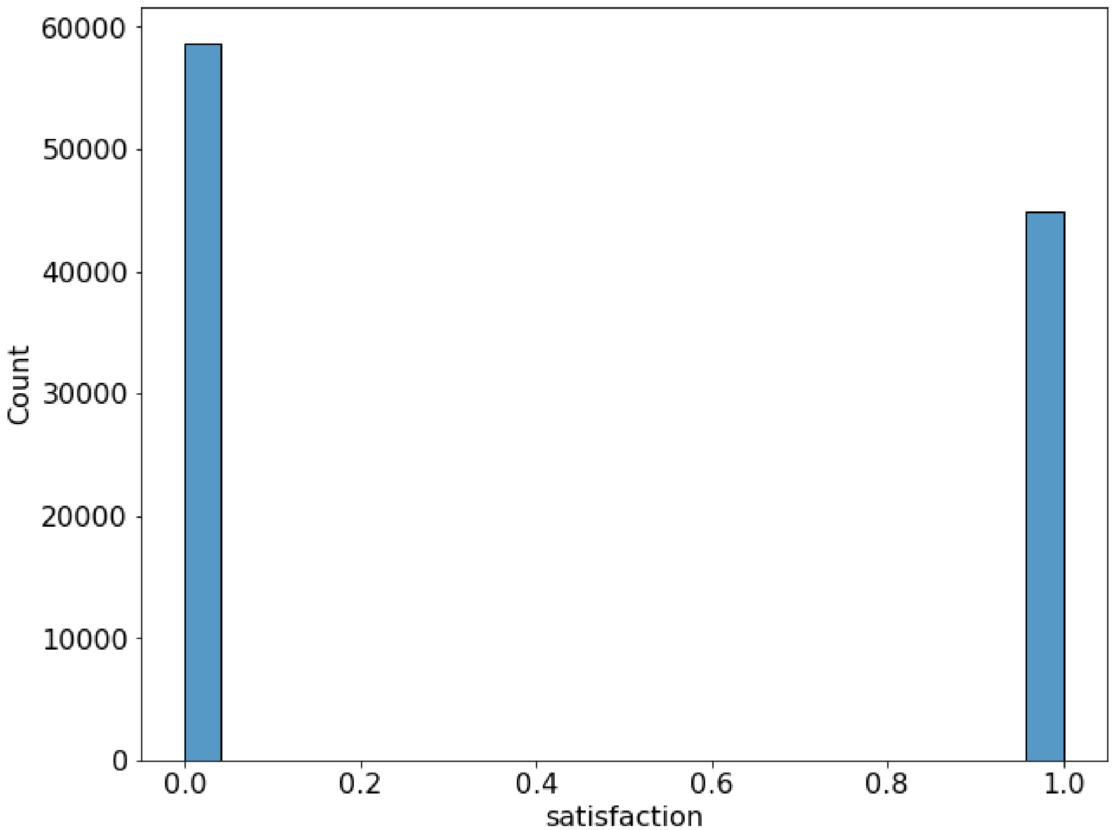
[223]:
```
# Zbiór treningowy dzielimy na treningowy i walidacyjny w proporcjach 9:1
y_aps = aps_train['satisfaction']
X_aps = aps_train.drop(['satisfaction'], axis = 1)
X_train_aps, X_val_aps, y_train_aps, y_val_aps = train_test_split(X_aps, y_aps,
 ↪random_state=420, test_size=0.1)

y_test_aps = aps_test['satisfaction']
X_test_aps = aps_test.drop(['satisfaction'], axis = 1)
```

[224]:
```
print("X Train : ", X_train_aps.shape)
print("X Val   : ", X_val_aps.shape)
print("X Test  : ", X_test_aps.shape)
print("Y Train : ", y_train_aps.shape)
print("Y Val : ", y_val_aps.shape)
print("Y Test  : ", y_test_aps.shape)
```

```
X Train :  (93234, 24)
X Val   :  (10360, 24)
X Test  :  (25893, 24)
Y Train :  (93234,)
Y Val :  (10360,)
Y Test  :  (25893,)
```

Rozkład zmiennej celu

[225]:
```
# Rozkład zmiennej celu
plt.figure(figsize=(10,8))
sns.histplot(aps_train['satisfaction'])
plt.show()
```



Zmienna dość zbalansowana - możemy spróbować użyć accuracy do oceny modelu.

Modelowanie

[240]:
```
# Random Forest Classifier Model
rfc=RandomForestClassifier()

rfc.fit(X_train_aps,y_train_aps)

print("accuracy score train : ", rfc.score(X_train_aps,y_train_aps))
print("accuracy score val : ", rfc.score(X_val_aps,y_val_aps))

y_predRFC=rfc.predict(X_val_aps)
y_predRFC_proba = rfc.predict_proba(X_val_aps)

print("r2 score val : " ,r2_score(y_val_aps,y_predRFC))
print("F1 score val : " ,f1_score(y_val_aps,y_predRFC))
print("RMSE score val : " ,sqrt(MSE(y_val_aps, y_predRFC)))
print("roc auc score val : " ,roc_auc_score(y_val_aps,y_predRFC))
```

```
accuracy score train :  1.0
accuracy score val :  0.9618725868725869
```

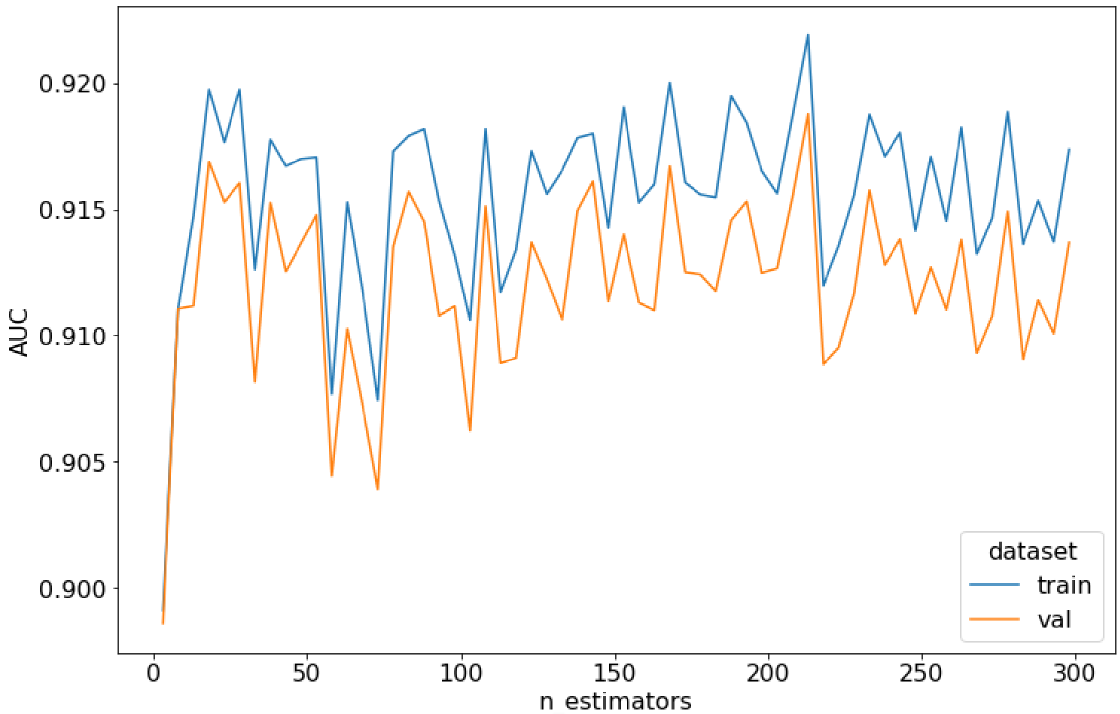
```
r2 score val :  0.845193198109467
F1 score val :  0.9559594157654143
RMSE score val :  0.19526242118598533
roc auc score val :  0.9597586486223417
```

Strojenie Random Forest AUC dla danych ilośći drzew w lesie z ograniczeniem na max głębokość drzewa = 5

[238]:
```
cols = ["n_estimators","AUC", "dataset"]
history = pd.DataFrame(columns=cols)

estimators = np.arange(3,300,5)
for trees_nr in tqdm(estimators):
    rf = RandomForestClassifier(n_estimators=trees_nr, max_depth=5, n_jobs=-1).
 ↪fit(X_train_aps, y_train_aps)
    train_score = roc_auc_score(y_train_aps, rf.predict(X_train_aps))
    val_score = roc_auc_score(y_val_aps, rf.predict(X_val_aps))
    history = history.append(dict(zip(cols, [trees_nr, train_score, "train"])),␣
 ↪ignore_index=True)
    history = history.append(dict(zip(cols, [trees_nr, val_score, "val"])),␣
 ↪ignore_index=True)
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=60.0),␣
 ↪HTML(value='')))
```

[239]:
```
sns.lineplot(data=history, x = "n_estimators", y = "AUC", hue = "dataset")
plt.show()
```



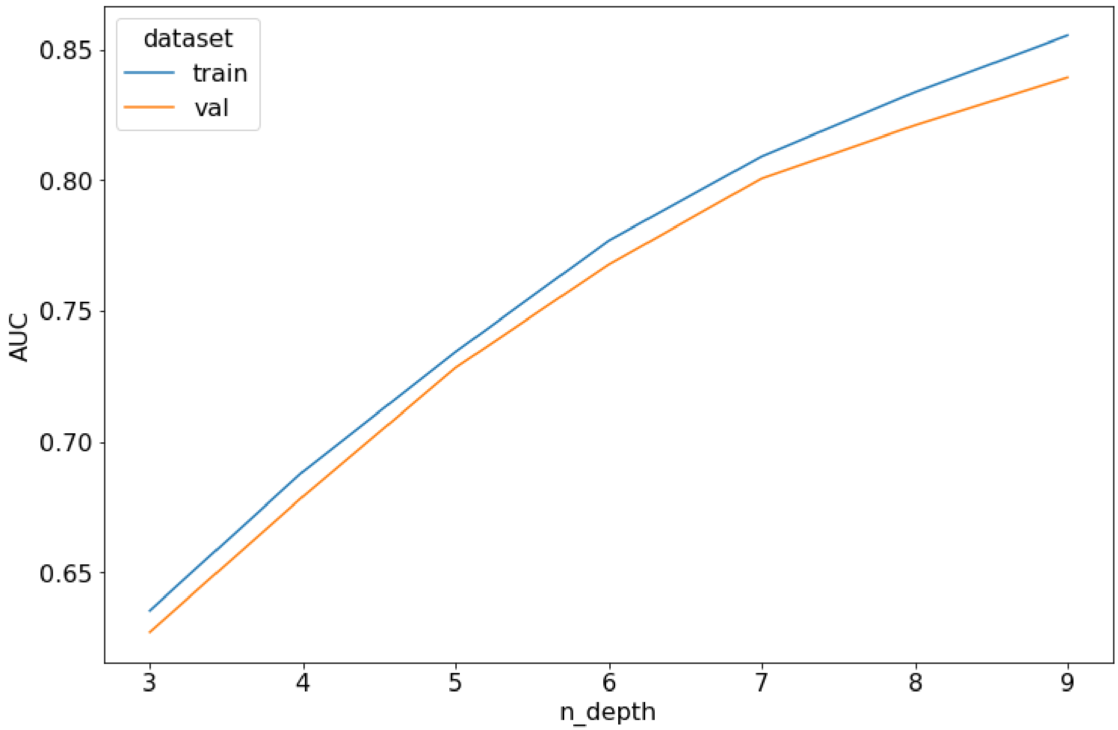
Dla dużej ilości drzew statystycznie lepsza generalizacja.

AUC dla danych głębokości drzew w lesie.

[297]:
```
cols = ["n_depth","AUC", "dataset"]
history = pd.DataFrame(columns=cols)

n_depth = np.arange(3,10,1)
for depth in tqdm(n_depth):
    rf = RandomForestRegressor(max_depth=depth, n_jobs=-1).fit(X_train_aps,␣
 ↪y_train_aps)
    train_score = rf.score(X_train_aps,y_train_aps)
    val_score = rf.score(X_val_aps,y_val_aps)
    history = history.append(dict(zip(cols, [depth, train_score, "train"])),␣
 ↪ignore_index=True)
    history = history.append(dict(zip(cols, [depth, val_score, "val"])),␣
 ↪ignore_index=True)
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=7.0),␣
 ↪HTML(value='')))
```

[298]:
```
sns.lineplot(data=history, x = "n_depth", y = "AUC", hue = "dataset")
plt.show()
```



[249]:
```
# XGBoost Classifier Model
xgb_cls = xgb.XGBClassifier(objective="binary:logistic", seed = 42,␣
↪use_label_encoder=False)

xgb_cls.fit(X_train, y_train, eval_metric="error")
#xgb_cls.fit(X_train_aps, y_train_aps, verbose=True, early_stopping_rounds=5,␣
↪eval_metric="error", eval_set=[(X_val_aps, y_val_aps)])

print("accuracy score train : ", xgb_cls.score(X_train_aps,y_train_aps))
print("accuracy score val : ", xgb_cls.score(X_val_aps,y_val_aps))

y_predXGB=xgb_cls.predict(X_val_aps)

print("r2 score val : " ,r2_score(y_val_aps,y_predXGB))
print("F1 score val : " ,f1_score(y_val_aps,y_predXGB))
print("RMSE score val : " ,sqrt(MSE(y_val_aps,y_predXGB)))
print("roc auc score val : " ,roc_auc_score(y_val_aps,y_predXGB))
```

```
accuracy score train :  0.9772078855353198
accuracy score val :  0.9648648648648649
r2 score val :  0.8573425926882177
F1 score val :  0.9596631205673759
RMSE score val :  0.18744368523675353
roc auc score val :  0.9634524375753735
```

XGBoost radzi sobie troszeczkę lepiej na zbiorze walidacyjnym - random forest przeuczony?

1.1.4 Regresja

[258]:
```
cpp = pd.read_csv("C:
↪\\Users\\pauli\\Downloads\\wb_hw2_data\\car_prices_poland\\Car_Prices_Poland.
↪csv")
```

[259]:
```
cpp.head()
```

[259]:
```
   Unnamed: 0  mark  model generation_name  year  mileage  vol_engine    fuel  \
0           0  opel  combo      gen-d-2011  2015   139568        1248  Diesel
1           1  opel  combo      gen-d-2011  2018    31991        1499  Diesel
2           2  opel  combo      gen-d-2011  2015   278437        1598  Diesel
3           3  opel  combo      gen-d-2011  2016    47600        1248  Diesel
4           4  opel  combo      gen-d-2011  2014   103000        1400     CNG

              city     province  price
0            Janki  Mazowieckie  35900
1         Katowice      Śląskie  78501
2            Brzeg     Opolskie  27000
3        Korfantów     Opolskie  30800
4  Tarnowskie Góry      Śląskie  35900
```

[260]:
```
cpp.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 117927 entries, 0 to 117926
Data columns (total 11 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   Unnamed: 0       117927 non-null  int64
 1   mark             117927 non-null  object
 2   model            117927 non-null  object
 3   generation_name  87842 non-null   object
 4   year             117927 non-null  int64
 5   mileage          117927 non-null  int64
 6   vol_engine       117927 non-null  int64
 7   fuel             117927 non-null  object
 8   city             117927 non-null  object
 9   province         117927 non-null  object
 10  price            117927 non-null  int64
dtypes: int64(5), object(6)
memory usage: 9.9+ MB
```

[261]:
```
cpp.isna().sum()
```

[261]:
```
Unnamed: 0              0
mark                    0
model                   0
generation_name     30085
year                    0
mileage                 0
vol_engine              0
fuel                    0
city                    0
province                0
price                   0
dtype: int64
```

Braki danych w kolumnie generation_name'

Kodowanie zmiennych kategorycznych

[262]:
```
# Label encoding dla zmiennej `mark`
LE=preprocessing.LabelEncoder()
LE.fit(cpp["mark"])
cpp["Mark"]=LE.transform(cpp["mark"])

# One hot encodind dla zmiennej `fuel`
encoded = pd.get_dummies(cpp[["fuel"]].astype(str))
encoded = encoded.drop(["fuel_CNG"], axis = 1)
cpp = pd.concat([cpp, encoded], axis = 1)

# pozostałe zmienne kategoryczne usuwam
cpp.drop(columns=["Unnamed: 0","mark","model","fuel","city","province","generation_name"],axis=1,inplace=True )#dropping
```

Podział zbioru na train, val i test

[263]:
```
y_cpp = cpp['price']
X_cpp = cpp.drop("price", axis = 1)
X_train_val_cpp, X_test_cpp, y_train_val_cpp, y_test_cpp = train_test_split(X_cpp, y_cpp, random_state=420, test_size=0.2)
X_train_cpp, X_val_cpp, y_train_cpp, y_val_cpp = train_test_split(X_train_val_cpp, y_train_val_cpp, random_state=420, test_size=0.125)
```

[264]:
```
print("X Train : ", X_train_cpp.shape)
print("X Val   : ", X_val_cpp.shape)
print("X Test  : ", X_test_cpp.shape)
print("Y Train : ", y_train_cpp.shape)
print("Y Val : ", y_val_cpp.shape)
print("Y Test  : ", y_test_cpp.shape)
```

```
X Train :  (82548, 9)
X Val  :  (11793, 9)
X Test  :  (23586, 9)
Y Train :  (82548,)
Y Val :  (11793,)
Y Test  :  (23586,)
```

Modelowanie

[289]:
```
# Random Forest Regressor Model
rfr = RandomForestRegressor()

rfr.fit(X_train_cpp,y_train_cpp)

print("Train r2 : ", rfr.score(X_train_cpp,y_train_cpp))
print("Val r2 : ", rfr.score(X_val_cpp,y_val_cpp))

y_predRFR=rfr.predict(X_val_cpp)

print("RMSE score val : " ,sqrt(MSE(y_val_cpp, y_predRFR)))
```

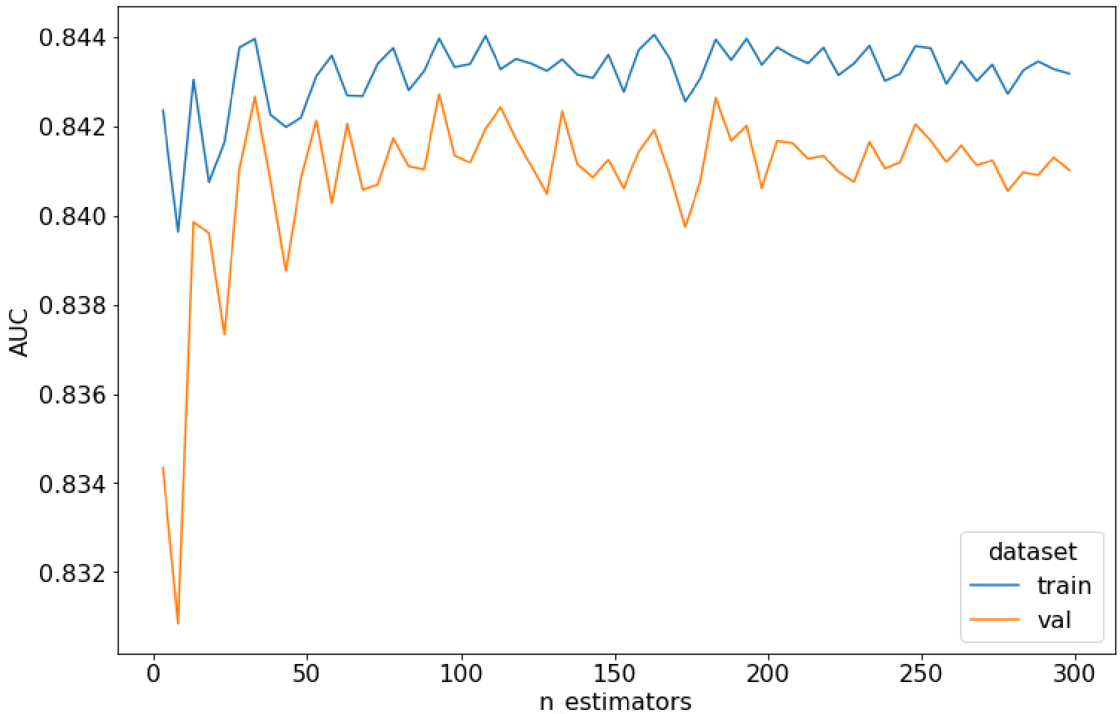
```
Train r2 :  0.9668626884290091
Val r2 :  0.9056295933202504
RMSE score val :  26522.960116519323
```

Strojenie Random Forest

[274]:
```
cols = ["n_estimators","AUC", "dataset"]
history = pd.DataFrame(columns=cols)

estimators = np.arange(3,300,5)
for trees_nr in tqdm(estimators):
    rf = RandomForestRegressor(n_estimators=trees_nr, max_depth=5, n_jobs=-1).
 ↪fit(X_train_cpp, y_train_cpp)
    train_score = rf.score(X_train_cpp,y_train_cpp)
    val_score = rf.score(X_val_cpp,y_val_cpp)
    history = history.append(dict(zip(cols, [trees_nr, train_score, "train"])),␣
 ↪ignore_index=True)
    history = history.append(dict(zip(cols, [trees_nr, val_score, "val"])),␣
 ↪ignore_index=True)
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=60.0),␣
 ↪HTML(value='')))
```

[275]:
```
sns.lineplot(data=history, x = "n_estimators", y = "AUC", hue = "dataset")
plt.show()
```

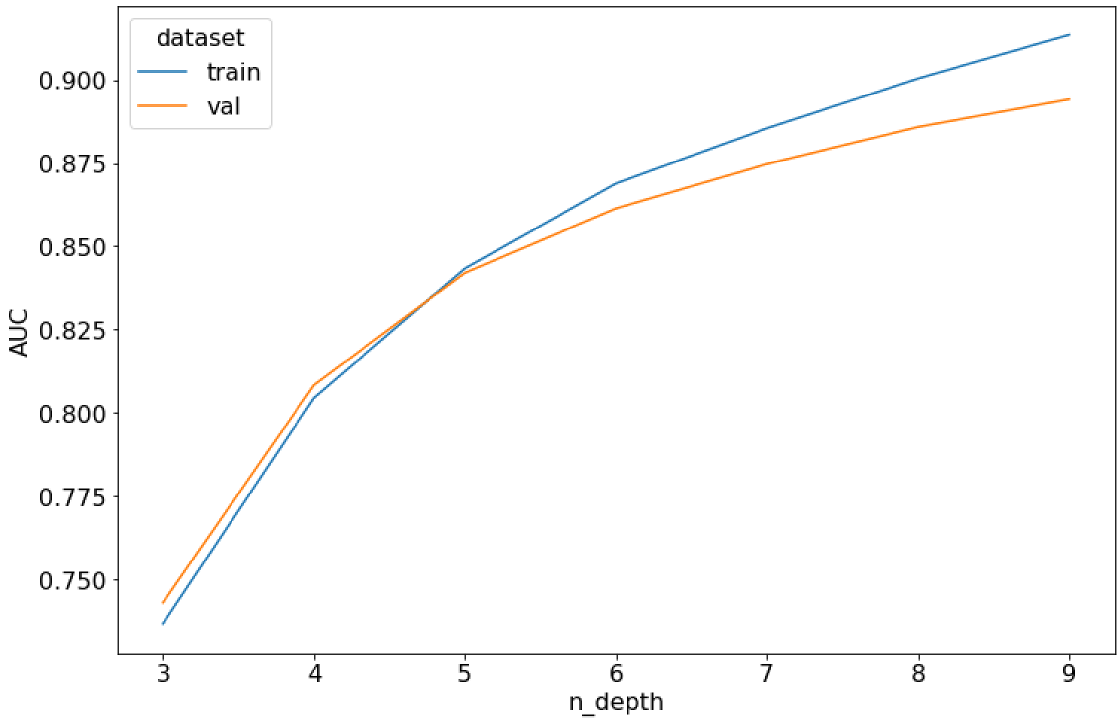


[296]:
```
cols = ["n_depth","AUC", "dataset"]
history = pd.DataFrame(columns=cols)

n_depth = np.arange(3,10,1)
for depth in tqdm(n_depth):
    rf = RandomForestRegressor(max_depth=depth, n_jobs=-1).fit(X_train_cpp,␣
 ↪y_train_cpp)
    train_score = rf.score(X_train_cpp,y_train_cpp)
    val_score = rf.score(X_val_cpp,y_val_cpp)
    history = history.append(dict(zip(cols, [depth, train_score, "train"])),␣
 ↪ignore_index=True)
    history = history.append(dict(zip(cols, [depth, val_score, "val"])),␣
 ↪ignore_index=True)
```

```
HBox(children=(HTML(value=''), FloatProgress(value=0.0, max=7.0),␣
 ↪HTML(value='')))
```

AUC dla danych głębokości drzew w lesie.

[295]:
```
sns.lineplot(data=history, x = "n_depth", y = "AUC", hue = "dataset")
plt.show()
```



Model zaczyna się przeuczać powyżej głębokości 5.

[286]:
```
# XGBoost Regressor Model
xgb_r = xg.XGBRegressor(objective ='reg:linear',
                  n_estimators = 10, seed = 123)

# Fitting the model
xgb_r.fit(X_train_cpp, y_train_cpp)

print("r2 score train : ", xgb_r.score(X_train_cpp,y_train_cpp))
print("r2 score val : ", xgb_r.score(X_val_cpp,y_val_cpp))

y_predXGB=xgb_r.predict(X_val_cpp)

print("RMSE score val : " ,sqrt(MSE(y_val_cpp,y_predXGB)))
```

```
[20:01:27] WARNING: C:/Users/Administrator/workspace/xgboost-
win64_release_1.5.1/src/objective/regression_obj.cu:188: reg:linear is now
deprecated in favor of reg:squarederror.
accuracy score train :  0.9000382648526956
accuracy score val :  0.8880666939284066
r2 score val :  0.8880666939284066
r2 score val :  0.9000382648526956
RMSE score val :  28885.756410297505
```