14 CZERWCA 2021

SPRAWOZDANIE PROJEKT

METODY I NARZĘDZIA BIG DATA

MAJA SZYMAJDA (254313)
INŻYNIERIA SYSTEMÓW ROK 2

SPIS TREŚCI

WPROWADZENIE

Jest to projekt realizowany na zajęcia laboratoryjne z przedmiotu: „Metody i Narzędzia Big Data”. Jego celem jest zapoznanie się z metodami oraz narzędziami służącymi rozwiązywaniu zadań na potrzeby Big Data i użycie ich do rozwiązania wybranego zadania z tej tematyki. Do realizacji tego projektu należało wykorzystać wiedzę i umiejętności nabyte podczas laboratoriów i wykładu z tego przedmiotu, a w razie konieczności wiedzę tę poszerzyć we własnym zakresie.

Projekt, który zrealizowałam dotyczy pojęcia klasyfikacji oraz predykcji modelu. Zbiór danych jaki wykorzystuje to „London bike sharing dataset”, który znalazłam na stornie: www.kaggle.com, jego dane są pozyskiwane z 3 źródeł: TfL, freemeteo.com, www.gov.uk/bank-holidays. Zawiera on informacje na temat wypożyczania rowerów w Londynie, w zależności od różnych czynników, takich jak pogoda, dzień tygodnia, miesiąc godzina itp.

ZBIÓR DANYCH

Zbiór danych zawiera informacje gromadzone od 1.01.2016 do 31.12.2017. W ciągu dwóch lat zebrano ogromną ilość danych, bo aż 8770 próbek, z każdego dnia i każdej godziny z podanego zakresu.

Rodzaje danych w zbiorze:

- "timestamp" - data pomiaru (rok-miesiac-dzień) i czas pormiaru (godzina-minuta-sekunda)
- "cnt" - liczba nowych osób wypożczających rowery
- "t1" - temperatura
- "t2" - temperatura odczuwalna
- "hum" - wilgotność powietrza
- "windspeed" - prędkość wiatru w km/h
- "weathercode" - kategoria pogody (opis poniżej)
- "is_holiday" - zmienna binarna - mówi o tym czy mamy okres wakacji czy nie 1 oznacza wakacje, 0 okres nie wakacyjny
- "is_weekend" - zmienna binarna - mówi o tym czy mamy weekend, 1 oznacza dni weekendowe, 0 pozostałe dni tyg
- "season" - oznacza pory roku: 0 - wiosna ; 1 - lato; 2 - jesień; 3 - zima.

Kategorie pogody:

- 1 = czyste niebo
- 2 = niewielkie zachmurzenie
- 3 = spore zachmurzenie
- 4 = zachmurzenie
- 7 = niewielkie opady
- 10 = opady i burze
- 26 = śnieżyca
- 94 = zamarzająca mgła

UŻYWANE NARZĘDZIA

Do realizacji tego projektu wykorzystałam język programowania Python i jako środowisko Notatnik Jupitera. Wykorzystałam również poniższe biblioteki:

- scipy
- pandas
- numpy
- sklearn
- seaborn
- warnings
- datetime
- matplotlib

SCHEMAT DZIAŁANIA

Lista kroków, które zaimplementowałam w projekcie:

1. Wczytanie danych
2. Analiza surowych danych (forma, typy danych, sprawdzenie ewentualnych braków, analiza podstawowych informacji statystycznych o wybranych danych)
3. Przygotowanie i przekształcenie danych (uzupełnienie ewentualnych braków w danych, usuwanie, dodawanie, rozbijanie kolumn)
4. Wizualizacja danych (sprawdzenie czy nie posiadamy braków w danych, wstępna analiza danych)
5. Wykrywanie wartości odstających, normalizacja i standaryzacja
6. Transformacja i selekcja cech (analiza składowych głównych – PCA, lub liniowa analiza dyskryminacyjna (LDA))
7. Dopasowanie modelu do danych (podział danych na uczące i testowe w proporcji 75:25, wybór modelu i uczenie go)
8. Ocena wyuczonego modelu przez porównanie statystyk modelu (precyzja, czułość, itp)

Implementacja projektu w języku Python

0. Import odpowiednich bibliotek:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import numpy.random as rnd
from datetime import datetime
from scipy import stats
from sklearn.decomposition import PCA
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC, SVR
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import VotingClassifier
from sklearn.metrics import precision_score, classification_report, roc_auc_score
from sklearn import datasets, metrics, model_selection, svm
import seaborn as sns
sns.set_style('whitegrid')
%matplotlib inline
```

1. Wczytanie danych:

```
filename = 'dane_rowery.csv'
df = pd.read_csv(filename)
print(df)
```

```
                timestamp   cnt   t1  ...  is_holiday  is_weekend  season
0     2016-01-01 00:00:00   786  6.0  ...         1.0         0.0     3.0
1     2016-01-01 01:00:00   660  5.5  ...         1.0         0.0     3.0
2     2016-01-01 02:00:00   387  5.0  ...         1.0         0.0     3.0
3     2016-01-01 03:00:00   294  5.0  ...         1.0         0.0     3.0
4     2016-01-01 04:00:00   219  5.0  ...         1.0         0.0     3.0
...                   ...   ...  ...  ...         ...         ...     ...
8766  2017-01-03 19:00:00  1042  5.0  ...         0.0         0.0     3.0
8767  2017-01-03 20:00:00   541  5.0  ...         0.0         0.0     3.0
8768  2017-01-03 21:00:00   337  5.5  ...         0.0         0.0     3.0
8769  2017-01-03 22:00:00   224  5.5  ...         0.0         0.0     3.0
8770  2017-01-03 23:00:00   139  5.0  ...         0.0         0.0     3.0

[8771 rows x 10 columns]
```

FIGURE 1 ZACZYTANIE DANYCH Z PLIKU CSV

2. Analiza surowych danych

```
df.head()
```

	timestamp	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season
0	2016-01-01 00:00:00	786	6.0	3.5	81.0	13.0	1.0	1.0	0.0	3.0
1	2016-01-01 01:00:00	660	5.5	3.0	84.0	11.0	1.0	1.0	0.0	3.0
2	2016-01-01 02:00:00	387	5.0	2.5	84.0	10.0	1.0	1.0	0.0	3.0
3	2016-01-01 03:00:00	294	5.0	4.5	81.0	5.0	1.0	1.0	0.0	3.0
4	2016-01-01 04:00:00	219	5.0	4.0	76.0	5.5	1.0	1.0	0.0	3.0

FIGURE 2 TABLICA ZAWIERAJĄCA DANE Z PLIKU

```
[4] df.describe()
```

	cnt	t1	t2	hum	wind_speed	weather_code	is_holiday	is_weekend	season
count	8771.000000	8771.000000	8771.000000	8771.000000	8771.000000	8771.000000	8771.000000	8771.000000	8771.000000
mean	1159.186638	12.318417	11.302284	73.272489	15.168130	2.696500	0.024627	0.286398	1.507240
std	1105.373438	5.907147	6.942326	14.217901	7.663815	2.292173	0.154993	0.452104	1.123619
min	9.000000	-1.000000	-5.000000	23.000000	0.000000	1.000000	0.000000	0.000000	0.000000
25%	248.000000	8.000000	5.000000	64.000000	9.000000	1.000000	0.000000	0.000000	1.000000
50%	856.000000	12.000000	12.000000	76.000000	14.000000	2.000000	0.000000	0.000000	2.000000
75%	1704.000000	16.500000	16.500000	84.000000	19.500000	3.000000	0.000000	1.000000	3.000000
max	5422.000000	33.000000	32.500000	100.000000	56.000000	26.000000	1.000000	1.000000	3.000000

FIGURE 3 PODSTAWOWE DANE STATYSTYCZNE

```
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8771 entries, 0 to 8770
Data columns (total 11 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   timestamp     8771 non-null   object
 1   cnt           8771 non-null   int64
 2   t1            8771 non-null   float64
 3   t2            8771 non-null   float64
 4   hum           8771 non-null   float64
 5   wind_speed    8771 non-null   float64
 6   weather_code  8771 non-null   float64
 7   is_holiday    8771 non-null   float64
 8   is_weekend    8771 non-null   float64
 9   season        8771 non-null   float64
 10  Id            8771 non-null   int64
dtypes: float64(8), int64(2), object(1)
memory usage: 753.9+ KB
```

FIGURE 4 INFORMACJA O TYPACH I WIELKOSCIACH DANYCH

3. Przygotowanie i przekształcenie danych:

```
df["Id"] = df.index + 1
```

```
df['timestamp'] = df['timestamp'] .apply(lambda x :datetime.strptime(x,'%Y-%m-%d %H:%M:%S'))
df['month'] = df['timestamp'].apply(lambda x : str(x).split(' ')[0].split('-')[1])
df['day'] = df['timestamp'].apply(lambda x : str(x).split(' ')[0].split('-')[2])
df['hour'] = df['timestamp'].apply(lambda x : str(x).split(' ')[1].split(':')[0])
```

FIGURE 5 DODANIE ID DO TABELI ORAZ PODZIAŁ TIMESTAMP NA MIESIAC, DATE, GODZINE

4. Wizualizacja danych:

```
fig,(ax1, ax2, ax3, ax4, ax5)= plt.subplots(nrows=5)
fig.set_size_inches(18,25)

sns.pointplot(data=df, x='hour', y='cnt', ax=ax1)
sns.pointplot(data=df, x='hour', y='cnt', hue='is_holiday', ax=ax2)
sns.pointplot(data=df, x='hour', y='cnt', hue='is_weekend', ax=ax3)
sns.pointplot(data=df, x='hour', y='cnt', hue='season', ax=ax4)
sns.pointplot(data=df, x='hour', y='cnt', hue='weather_code',ax=ax5)
```

FIGURE 6 WYŚWIETLENIE DANYCH O ILOŚCI WYPOZYCZONYCH ROWERÓW W ZALEŻNOŚCI OD GODZINY

```
[20] for i in numerical_df:
       if i == "data" or i == 'Id':
         continue
       else:
         sns.displot(x = i, data = xdf, height = 6, aspect = 2, kde = True);
         plt.xlabel(i, fontsize = 15)
```

FIGURE 7 WYŚWIETLENIE DYSTRYBUCJI WSZYSTKICH DANYCH W ZALEŻNOŚCI OD LICZBY WYPOŻYCZONYCH ROWERÓW

```
for i in numerical_df:
  if i == "data" or i == 'Id':
    continue
  else:
    plt.figure(figsize = (8,6))
    sns.scatterplot(x = i, y = "Id", data = xdf);
    plt.xlabel(i, fontsize = 12)
    plt.ylabel("Id", fontsize = 12)
```

FIGURE 8 WYŚWIETLENIE WYKRESU ROZPROSZENIA DANYCH

5. Wykrywanie wartości odstających, normalizacja i standaryzacja

```
[15] def sqrt_transformation(dataframe):
       return np.sqrt(dataframe)


     # function for removing outliers

     def remove_outliers(dataframe, column):
       q3 = dataframe[column].quantile(0.75)
       q1 = dataframe[column].quantile(0.25)
       IQR = q3 - q1

       upper = q3 + (1.5 * IQR)
       lower = q1 - (1.5 * IQR)

       dataframe = dataframe[(dataframe[column] > lower ) & (dataframe[column] < upper)]

       return dataframe
```

```
[16] xdf = remove_outliers(xdf,'cnt')
```

FIGURE 9 PRZYKŁADOWA NORMALIZACJA DLA CNT

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X1 = scaler.fit_transform(X)
X = pd.DataFrame(data = X1, columns = X.columns)

X_b_PCA = xdf
```

FIGURE 10 STANDARYZACJA Z WYKORZYSTANIEM BIBLIOTEKI SKLEARN

6. Transformacja i selekcja cech:

```
pca = PCA(n_components=4)
X = pca.fit_transform(X_b_PCA)
print(pca.explained_variance_ratio_)
print()
print(pca.singular_values_)

print(X_b_PCA.shape)
print(X.shape)
print(X)
```

```
[9.99589917e-01 1.97865924e-04 7.92436383e-05 6.44330874e-05]

[83360.98556266  1172.83633122   742.22249167   669.27773619]
(8435, 12)
(8435, 4)
[[-2.43283120e+02  8.55407483e+00  5.76614121e+00 -3.91128077e+00]
 [-3.69302714e+02  1.07943237e+01  5.15362990e+00 -4.96061874e+00]
 [-6.42292658e+02  8.78135443e+00  5.23393430e+00 -5.93122557e+00]
 ...
 [-6.92178457e+02 -8.14158312e-01  1.44363004e+01  6.37775267e+00]
 [-8.05153468e+02 -3.99936872e+00  1.44834906e+01  4.87647988e+00]
 [-8.90150215e+02 -4.49448969e+00  1.47586144e+01  3.64515580e+00]]
```

FIGURE 11 PCA DLA N = 4

7. Dopasowanie modelu do danych:

```
[46] X_train, X_test, Y_train, Y_test = train_test_split(X, Y)

     X_train.shape, X_test.shape, Y_train.shape, Y_test.shape

     ((6326, 4), (2109, 4), (6326,), (2109,))
```

8. Ocena wyuczonego modelu:

```
[34] clf1 = KNeighborsClassifier(n_neighbors=5)
     clf1.fit(X_train, Y_train)

     predictArr = list(clf1.predict(X_test))

     precScore = precision_score(Y_test, predictArr, average='macro')

     classReport = classification_report(Y_test, predictArr)

     print(precScore)
     print(classReport)

     0.45550900083449575
                   precision    recall  f1-score   support

              0.0       0.40      0.42      0.41       569
              1.0       0.54      0.71      0.61       504
              2.0       0.31      0.25      0.28       480
              3.0       0.58      0.47      0.52       556

         accuracy                           0.46      2109
        macro avg       0.46      0.46      0.45      2109
     weighted avg       0.46      0.46      0.46      2109
```

FIGURE 12 METODA KNN - K NAJBLIŻSZYCH SĄSIADÓW

```
[35] clf2 = make_pipeline(StandardScaler(), SVC(gamma='auto', probability=True))
     clf2.fit(X_train, Y_train)
     predictArr = list(clf2.predict(X_test))
     precScore = precision_score(Y_test, predictArr, average='macro')

     classReport = classification_report(Y_test, predictArr)

     print(precScore)
     print(classReport)

     0.6003103416661235
                   precision    recall  f1-score   support

              0.0       0.70      0.34      0.45       569
              1.0       0.65      0.92      0.76       504
              2.0       0.47      0.32      0.38       480
              3.0       0.59      0.84      0.69       556

         accuracy                           0.60      2109
        macro avg       0.60      0.60      0.57      2109
     weighted avg       0.60      0.60      0.57      2109
```

FIGURE 13 METODA SVM - MASZYNA WEKTORÓW NOŚNYCH

```
[36] clf3 = DecisionTreeClassifier()
     clf3.fit(X_train, Y_train)

     predictArr = list(clf3.predict(X_test))
     precScore = precision_score(Y_test, predictArr, average='macro')

     classReport = classification_report(Y_test, predictArr)
     print(precScore)
     print(classReport)

     0.5433956866864631
                   precision    recall  f1-score   support

              0.0       0.50      0.47      0.49       569
              1.0       0.68      0.68      0.68       504
              2.0       0.37      0.42      0.40       480
              3.0       0.62      0.59      0.60       556

         accuracy                           0.54      2109
        macro avg       0.54      0.54      0.54      2109
     weighted avg       0.55      0.54      0.54      2109
```

FIGURE 14 METODA DRZEW DECYZYJNCH

```
[37] clf4 = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 2), random_state=1)
     clf4.fit(X_train, Y_train)

     predictArr = list(clf4.predict(X_test))

     precScore = precision_score(Y_test, predictArr, average='macro')

     classReport = classification_report(Y_test, predictArr)
     print(precScore)
     print(classReport)
```

```
0.13269484698056128
              precision    recall  f1-score   support

         0.0       0.00      0.00      0.00       569
         1.0       0.31      0.55      0.40       504
         2.0       0.22      0.55      0.31       480
         3.0       0.00      0.00      0.00       556

    accuracy                           0.26      2109
   macro avg       0.13      0.28      0.18      2109
weighted avg       0.12      0.26      0.17      2109
```

FIGURE 15 SIEĆ NEURONOWA

```
[38] eclf = VotingClassifier(estimators=[('knn', clf1), ('svc', clf2), ('dt', clf3), ('nrn', clf3)], voting='hard')
     eclf = eclf.fit(X_train, Y_train)
     predictArr = list(eclf.predict(X_test))

     precScore = precision_score(Y_test, predictArr, average='macro')

     classReport = classification_report(Y_test, predictArr)
     print(precScore)
     print(classReport)
```

```
0.5495799117653867
              precision    recall  f1-score   support

         0.0       0.50      0.49      0.50       569
         1.0       0.65      0.82      0.73       504
         2.0       0.39      0.34      0.37       480
         3.0       0.65      0.59      0.62       556

    accuracy                           0.56      2109
   macro avg       0.55      0.56      0.55      2109
weighted avg       0.55      0.56      0.55      2109
```

FIGURE 16 WSZYSTKIE METODY

```
[39] from sklearn.linear_model import LogisticRegression
     clf = LogisticRegression(solver="liblinear").fit(X_train, Y_train)

     x = clf.predict_proba(X_test)
     y = Y_test

     print(x, y)

     roc_auc_score(Y_test, clf.predict_proba(X_test), multi_class='ovr')
```

```
[[0.0274971  0.60984321 0.35383078 0.00882892]
 [0.24265781 0.038645   0.20971563 0.50898156]
 [0.06927722 0.54643625 0.35849807 0.02578847]
 ...
 [0.34304285 0.01796664 0.14752149 0.49146902]
 [0.39568842 0.00107713 0.12054986 0.48268459]
 [0.04101301 0.58108624 0.32385832 0.05404243]] 5902    2.0
857     3.0
6230    2.0
5848    2.0
3085    0.0
       ...
711     3.0
5027    1.0
7442    2.0
374     3.0
5101    1.0
Name: season, Length: 2109, dtype: float64
0.8044700857653033
```

FIGURE 17 REGRESJA LOGISTYCZNA

9. Graficzna reprezentacja modelu

```
plt.scatter(X_train[:,0], X_train[:,1])
plt.title('Dane liniowo separowalne')
plt.xlabel('X_train')
plt.ylabel('X_train')
plt.show()
```

FIGURE 18 WYKRES DLA CIĄGU UCZĄCEGO

```
matrix = plot_confusion_matrix(clf2, X_test, Y_test, cmap=plt.cm.Blues, normalize='true')
plt.title('Confusion matrix dla naszego klasyfikatora')
plt.show(matrix)
plt.show()
```

FIGURE 19 GENEROWANIE TABLICY POMYŁEK

```
clf = svm.SVC(kernel='linear')
clf = clf.fit(X_train, Y_train)

predictions = clf.predict(X_test)
support_vectors = clf.support_vectors_


plt.scatter(X_train[:,0], X_train[:,1])
plt.scatter(support_vectors[:,0], support_vectors[:,1], color='red')
plt.title('Dane liniowo separowalne wraz z support vectors')
plt.xlabel('X_tren')
plt.ylabel('X_tren')
plt.show()
```

FIGURE 20 WYKRES DLA MACIERZY WEKTORÓW NOŚNYCH

ANALIZA DANYCH

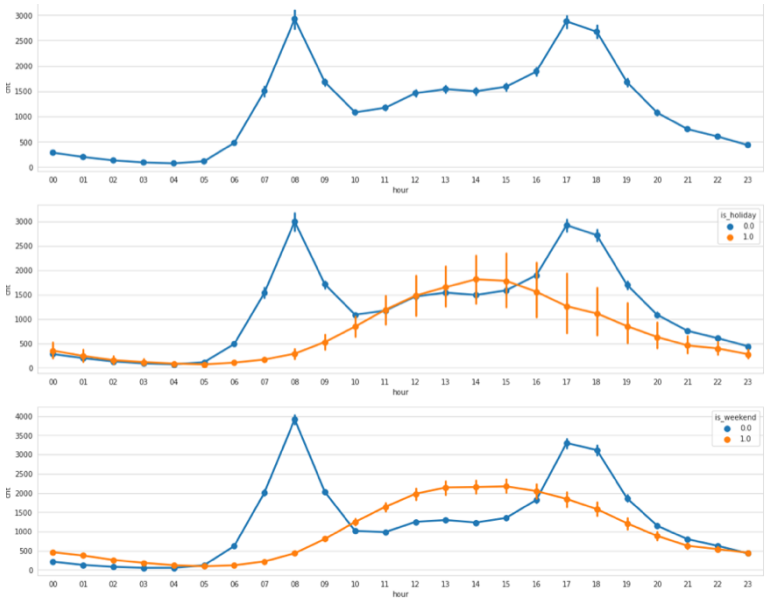
Dzięki opisanej wcześniej implementacji możemy przeanalizować surowe dane.

Legenda:

- "timestamp" - data pomiaru (rok-miesiac-dzień) i czas pormiaru (godzina-minuta-sekunda)
- "cnt" - liczba nowych osób wypożczających rowery
- "t1" - temperatura
- "t2" - temperatura odczuwalna
- "hum" - wilgotność powietrza
- "windspeed" - prędkość wiatru w km/h
- "weathercode" - kategoria pogody (opis poniżej)
- "is_holiday" - zmienna binarna - mówi o tym czy mamy okres wakacji czy nie 1 oznacza wakacje, 0 okres nie wakacyjny
- "is_weekend" - zmienna binarna - mówi o tym czy mamy weekend, 1 oznacza dni weekendowe, 0 pozostałe dni tyg
- "season" - oznacza pory roku: 0 - wiosna ; 1 - lato; 2 - jesień; 3 - zima.

Kategorie pogody:

- 1 = czyste niebo
- 2 = niewielkie zachmurzenie
- 3 = spore zachmurzenie
- 4 = zachmurzenie
- 7 = niewielkie opady
- 10 = opady i burze
- 26 = śnieżyca
- 94 = zamarzająca mgła

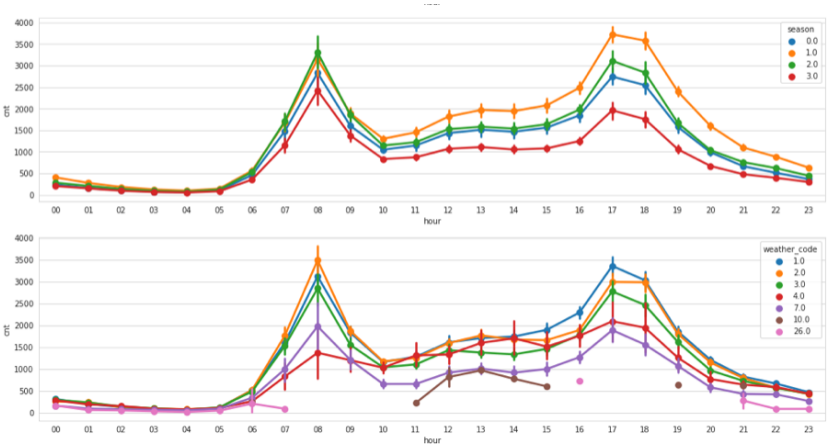


Pierwsze trzy wykresy reprezentują jak rozkładają się dane ilość nowych wypożyczeń rowerów do czasu.

Pierwszy zawiera dane o średnim rozkładzie cnt do godziny.

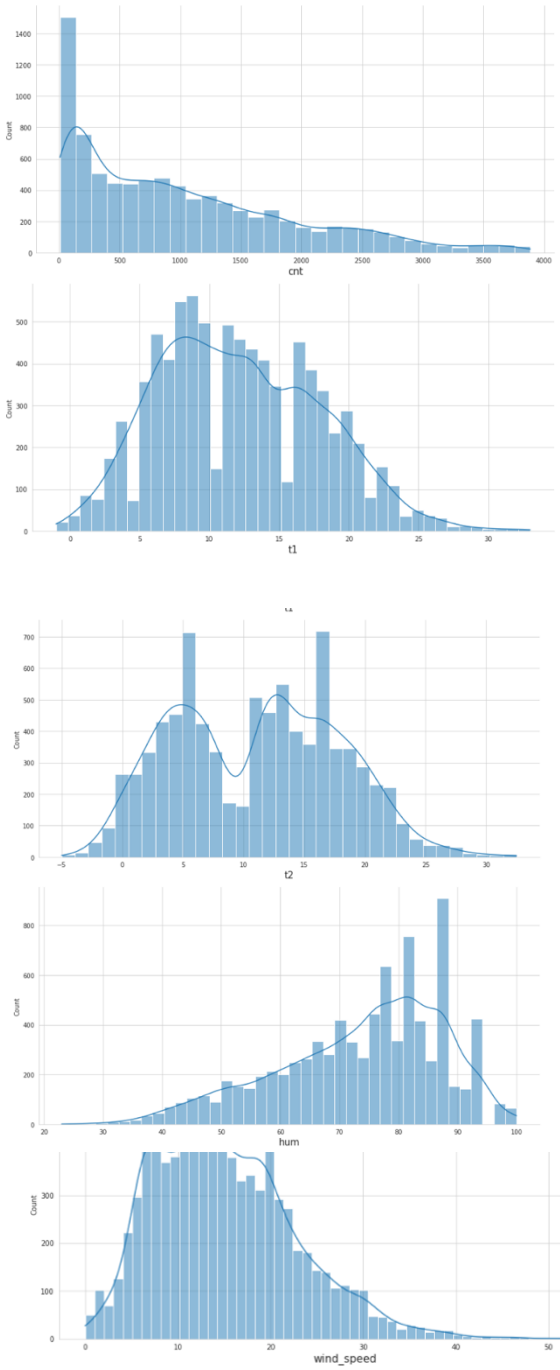
Drugi wykres ma pokazuje, jak rozkładają się dane z podziałem na wakacje i rok szkolny. (0 – rok szkolny; 1 – wakacje).

Trzeci wykres ma pokazuje, jak rozkładają się dane z podziałem na wakacje i rok szkolny. (0 – dni robocze; 1 – weekendy).

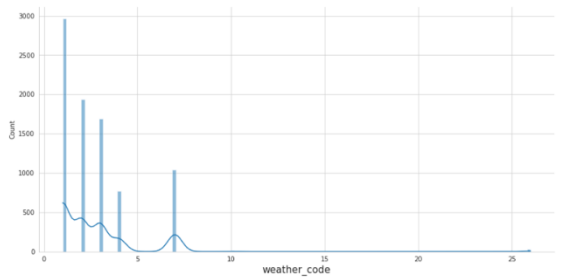


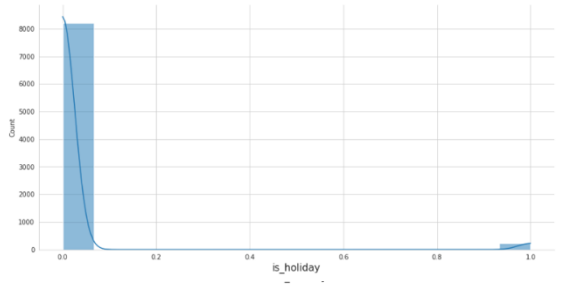
Kolejne dwa wykresy reprezentują ilość cnt w zależności od godziny podzielone ze względu na porę roku (0 – wiosna; 1 – lato; 2 – jesień; 3 – zima).

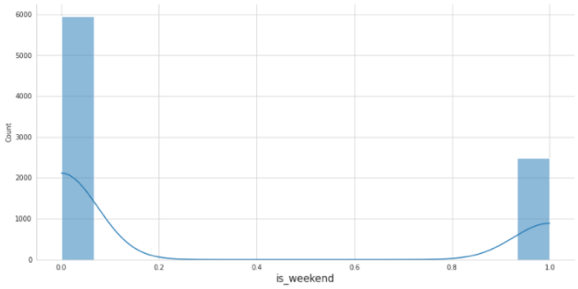
Drugi zawiera te dane z podziałem na kategorie pogody wymienione powyżej.

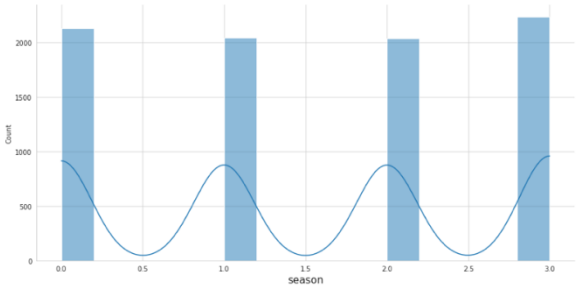


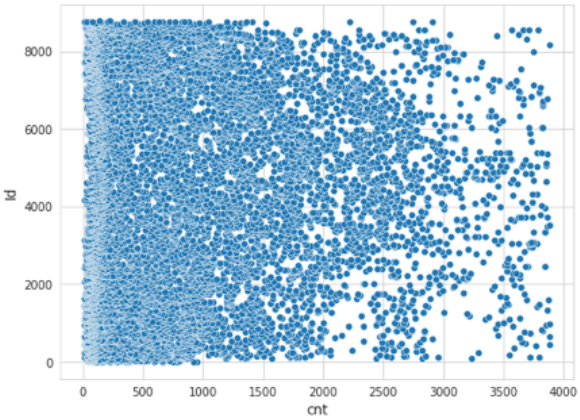
Kolejna seria wykresów dystrybucji, reprezentują liczby danych o danej wielkości np. najwięcej wartości cnt jest 0 i jest ich ponad 1400. Widać na tych wykresach, że posiadamy wartości odstające. Więc musimy poddać dane normalizacji i standaryzacji.

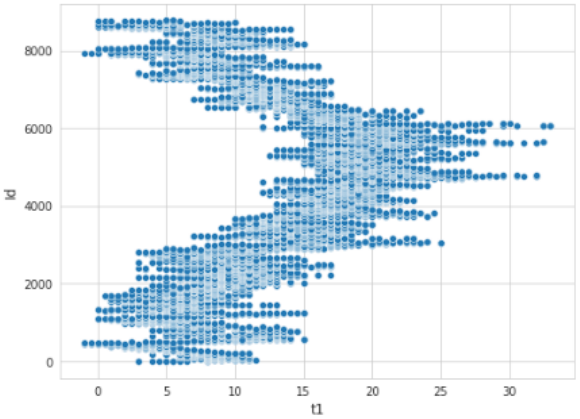






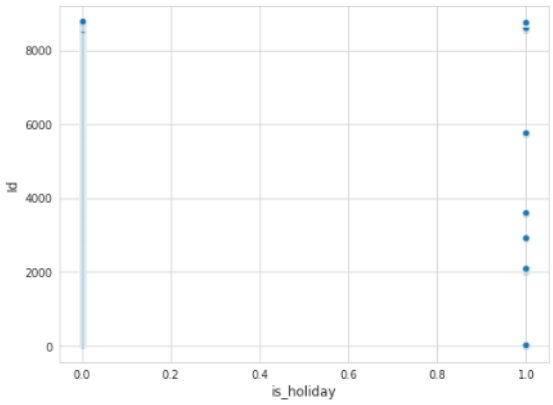


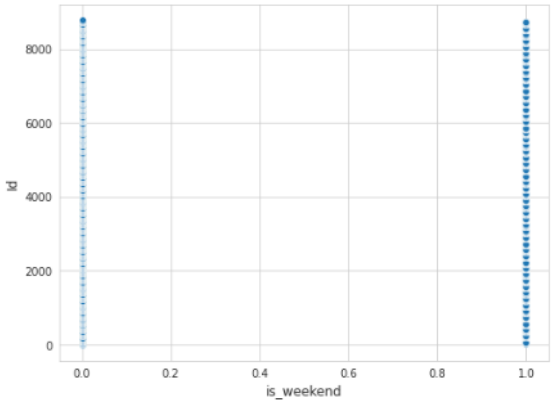


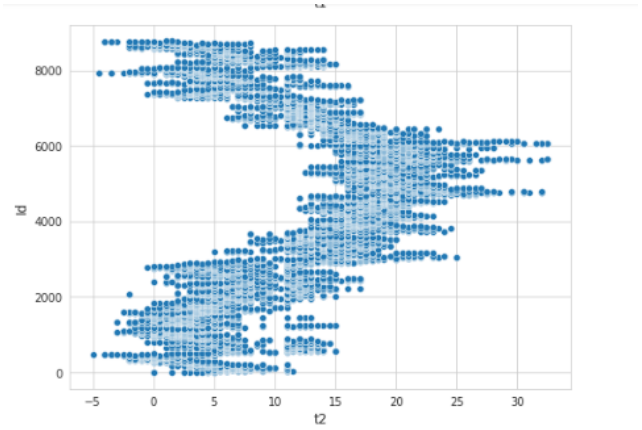


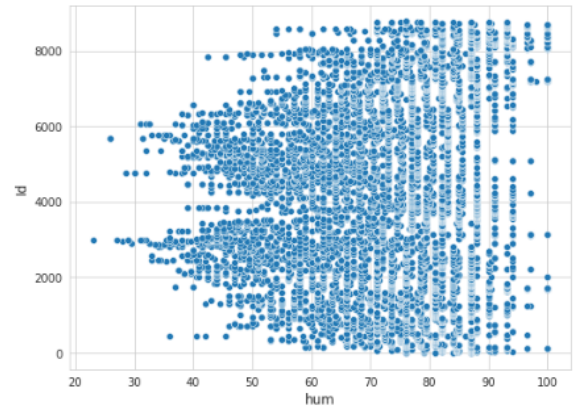
Wykres punktowy (ang. scatter plot), zwany także m.in. wykresem rozproszenia, rozrzutu, czy też punktowym diagramem korelacji, tworzony jest z dwóch do czterech zmiennych w postaci miar (measures) lub większej ilości wymiarów (dimensions). Pierwsze dwie główne miary, między którymi chcemy przetestować zależność, tworzą osie x i y, zaś kolejne miary i/lub wymiary mogą być użyte do tworzenia kontekstu dla punktów wykresu.

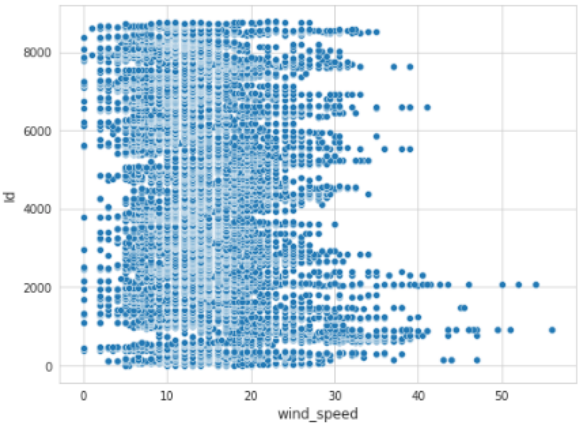
Zaprezentowane po prawej stronie i na stronie następnej, są wykresami dwuwymiarowymi, które przedstawiają rozkład danych w zależności od poszczególnych danych.

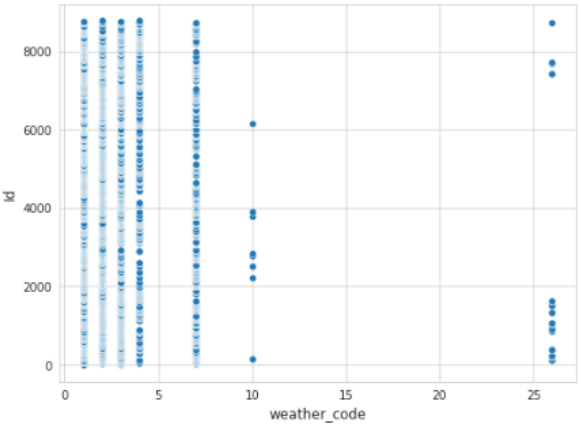

cnt
is_holiday

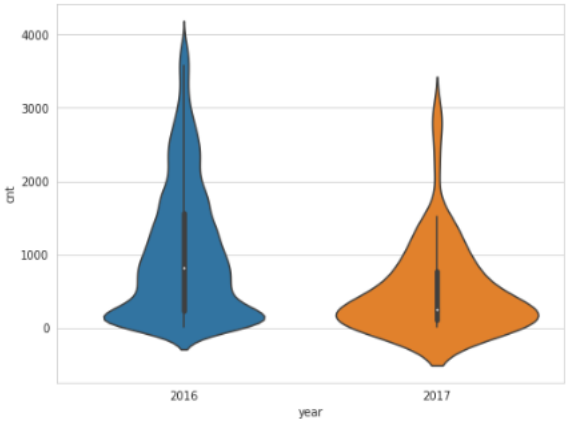

cnt
is_weekend

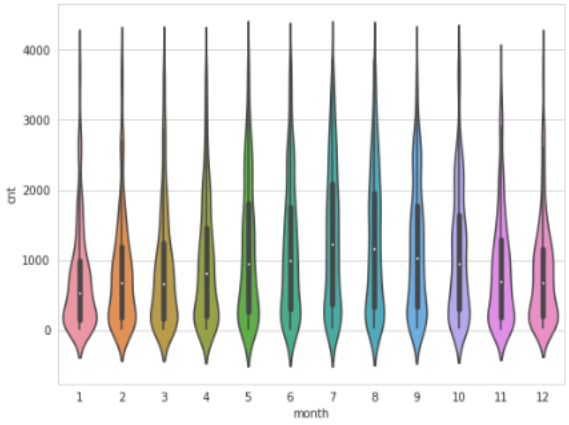

cnt
t2

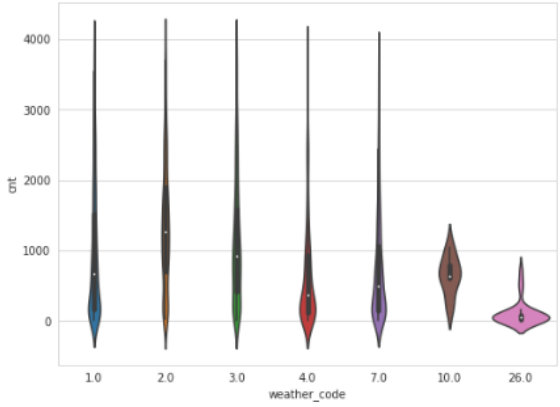

cnt
hum

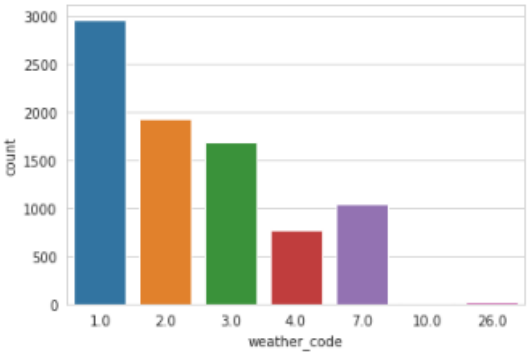

cnt
wind_speed


cnt
weather_code









Po lewej stronie umiszczone zostały przykłady wykresu skrzypcowego, które prezentują zależności ilości wypożyczonych rowerów od roku czy miesiąca a nawet różnych kategorii pogodowych.

Każde „skrzypce” przedstawiają grupę lub zmienną. Kształt skrzypiec reprezentuje tak naprawdę oszacowanie gęstości zmiennej: im więcej punktów danych w określonym zakresie, tym większe skrzypce będą w tym miejscu.

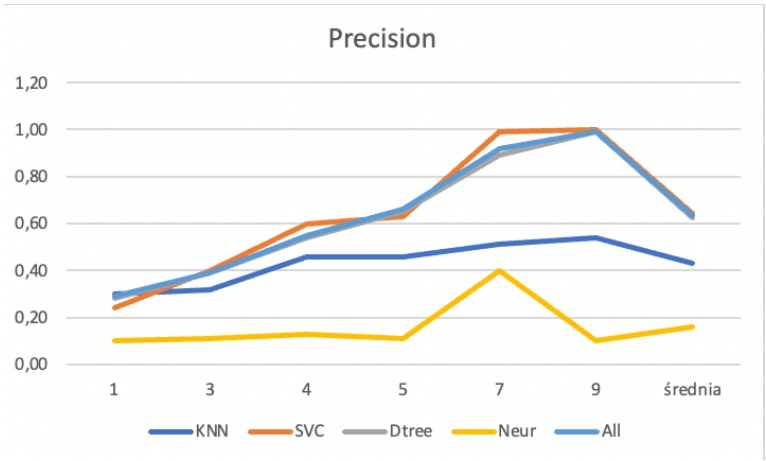
Skrzypce są szczególnie przystosowane, gdy ilość danych jest ogromna, a pokazanie indywidualnych obserwacji staje się niemożliwe. W przypadku małych zestawów danych lepszym rozwiązaniem jest prawdopodobnie wykres pudełkowy, ponieważ pokaże wszystkie informacje.

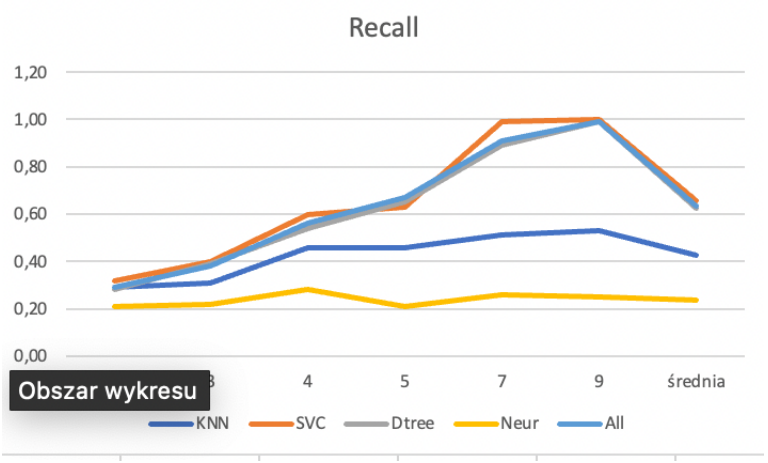
Jak widać na wykresach najwięcej danych mamy dla wartości ctn = 0.

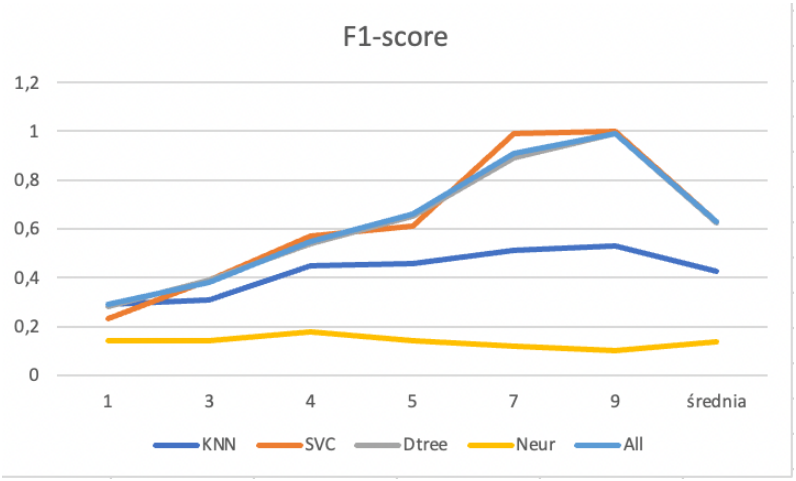
WYNIKI

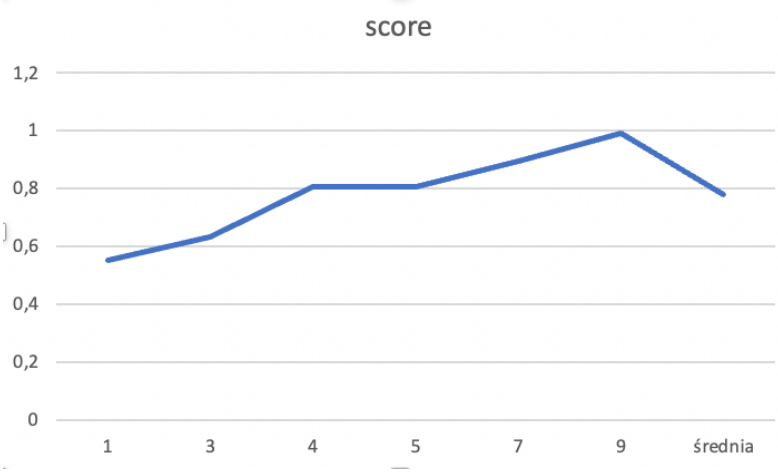
Wyniki, które otrzymałam różnią się w zależności od parametrów oraz modeli. O to wyniki dla części z nich. Oraz tabele z wartościami dla wszystkich prób, które przeprowadziłam.

Precision							
n	1	3	4	5	7	9	średnia
KNN	0,30	0,32	0,46	0,46	0,51	0,54	0,43
SVC	0,24	0,40	0,60	0,63	0,99	1,00	0,64
Dtree	0,28	0,39	0,54	0,65	0,89	0,99	0,62
Neur	0,10	0,11	0,13	0,11	0,40	0,10	0,16
All	0,29	0,39	0,55	0,66	0,92	0,99	0,63
Recall							
n	1	3	4	5	7	9	średnia
KNN	0,29	0,31	0,46	0,46	0,51	0,53	0,43
SVC	0,32	0,40	0,60	0,63	0,99	1,00	0,66
Dtree	0,28	0,39	0,54	0,65	0,89	0,99	0,62
Neur	0,21	0,22	0,28	0,21	0,26	0,25	0,24
All	0,29	0,38	0,56	0,67	0,91	0,99	0,63
F1-score							
n	1	3	4	5	7	9	średnia
KNN	0,29	0,31	0,45	0,46	0,51	0,53	0,425
SVC	0,23	0,39	0,57	0,61	0,99	1	0,63166667
Dtree	0,28	0,39	0,54	0,65	0,89	0,99	0,62333333
Neur	0,14	0,14	0,18	0,14	0,12	0,1	0,13666667
All	0,29	0,38	0,55	0,66	0,91	0,99	0,63
Roc Auc Score							
n	1	3	4	5	7	9	średnia
score	0,553	0,631	0,804	0,805	0,896	0,99	0,77983333



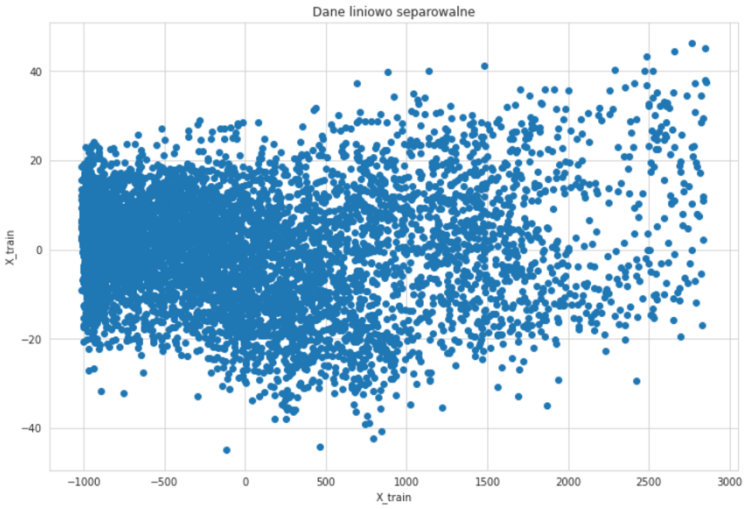


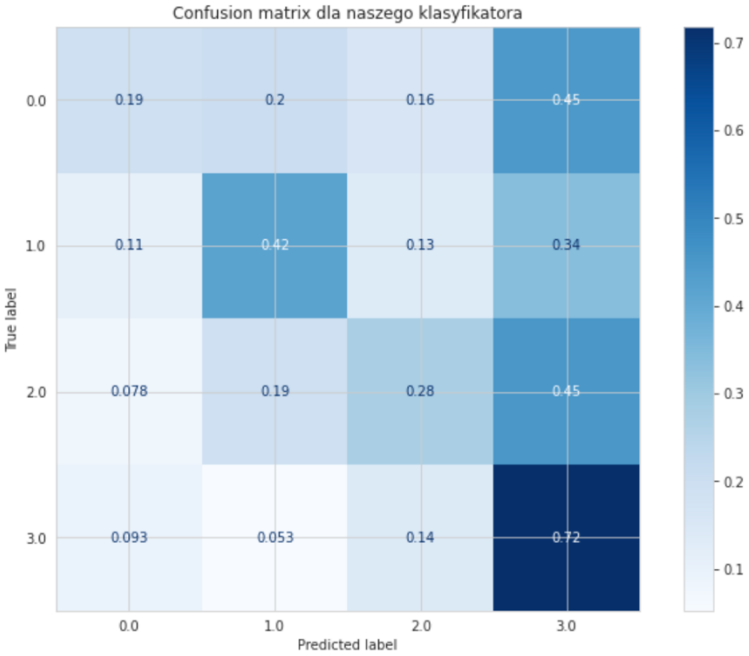


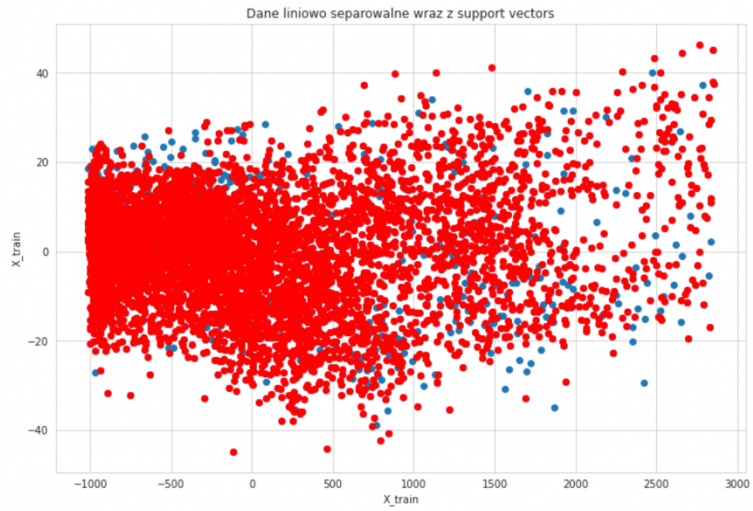


Jak widać wraz ze wzrostem n możemy zauważyć lepszą dokładność klasyfikacji.

Wykresy dla SVC dla n=3:







PODSUMOWANIE

Posumowanie rozpocznę od metody KNN – metoda k najbliższych sąsiadów, w naszym przypadku 5. Z analizy kryterium Precision, które reprezentuje dokładność rozpoznania w obrębie klasy możemy stwierdzić, że nasz model jest najbardziej skuteczny dla n=9, czyli dziewięciu wektorów własnych stworzonych w metodzie PCA. Osiąga wtedy ponad 50% skuteczność w klasyfikacji. Porównując jednak do pozostałych wyników, nie ma znacznej różnicy między nimi. Jeżeli chodzi o Recall, który odpowiada za rozpoznanie ilości elementów z klasy, to osiąga skuteczność 53%, czyli niewiele więcej niż co druga z pór roku zostaje właściwie określona. F1 score to metryka klasyfikatora, która uśrednia uzyskane dane licząc średnią harmoniczną z dwóch poprzednich.

Metoda SVC - Maszyna wektorów nośnych, maszyna wektorów podpierających (ang. Support Vector Machine, SVM), osiągnęła 100% skuteczność w dokładności rozpoznawania w obrębie danej klasy. Dawała ona satysfakcjonujące wyniki już przy 8 wektorach własnych, co widać w każdej z trzech miar skuteczności metody. Jeżeli chodzi o rozpoznanie ilości elementów to również nie można nic zarzucić tej metodzie.

Kolejną metodą było drzewo decyzyjne, które najlepszą skuteczność osiągnęło przy n=9. Niemal w 100% dobrze klasyfikował porę roku, w zależności od pozostałych danych. Jeżeli chodzi o rozpoznanie ilości elementów – metrykę Recall to jest ona również skuteczna w 99%.

Ostatnią metodą klasyfikacji była sieć neuronowa. Jej skuteczność w porównaniu do pozostałych modeli wypadła ta metoda najgorzej, średnia jego skuteczność wyniosła niewiele ponad 13%. Najlepsze wyniki osiągnęła dla n = 4. Recall w tym przypadku wynosi zaledwie 28%. Dla porównania dla n=4 następny najgorszy wynik osiągnęła metoda KNN z wynikiem 45% co jest zdecydowanie lepszym wynikiem. Jednak jego słaba skuteczność może wynikać z powodu niewystarczającej liczby danych, było ich ponad 8 tysięcy.

WNIOSKI

Podsumowując, najlepsze rezultaty w rozpoznawaniu klas osiągnęłam metodą SVC. Jednak w związku, z tym że jest to kosztowny i długotrwały proces (wykonywał się od 2 do aż 10 min), to w doborze klasyfikatora kierowałabym się jednak w stronę drzewa decyzyjnego. Jest ono niewiele mniej skuteczne a wykonuje się zdecydowanie krócej w porównaniu do maszyny wektorów nośnych.

Natomiast jeżeli chodzi o ilość wektorów własnych oznaczanych jako n, to najlepsze wyniki osiągamy dla n=9, co widać na wykresach powyżej.

Konieczne zgody i licencje:

'CONTAINS OS DATA © CROWN COPYRIGHT AND DATABASE RIGHTS 2016'