

ROZPOZNAWANIE CHOROBY COVID-19 NA ZDJĘCIACH RENTGENOWSKICH PŁUC Z WYKORZYSTANIEM UCZENIA MASZYNOWEGO

RECOGNITION OF COVID-19 DISEASE
FROM X-RAY CHEST IMAGES
WITH APPLICATION OF MACHINE LEARNING



AUTOR

Łukasz Więckowski

PROMOTOR

dr hab. inż., prof. ZUT Przemysław Klęsk
Katedra Metod Sztucznej Inteligencji i Matematyki Stosowanej

PLAN PREZENTACJI

1. **CELE PRACY**
2. **WYBÓR TEMATU PRACY**
3. **TEZA/HIPOTEZA PRACY**
4. **ZADANIA PRACY**
5. **SPIS TREŚCI**
6. **METODY BADAWCZE**
7. **ZAWARTOŚĆ CZĘŚCI TEORETYCZNEJ**
8. **ZAWARTOŚĆ CZĘŚCI PRAKTYCZNEJ**
9. **TRUDNOŚCI W REALIZACJI PRACY**
10. **STOPIEŃ ZAAWANSOWANIA REALIZACJI ZADAŃ PRACY**
11. **HARMONOGRAM REALIZACJI ZADAŃ PRACY**
12. **BIBLIOGRAFIA**

CELE PRACY

Celem części **praktycznej** pracy jest przeprowadzenie badań porównawczych nad dokładnością rozpoznawania choroby COVID-19 (na podstawie zdjęć rentgenowskich płuc) z wykorzystaniem różnych algorytmów uczenia maszynowego, w szczególności: głębokich sieci neuronowych, boostingu, algorytmu SVM oraz klasyfikatorów Bayesowskich.

Celem części **teoretycznej** pracy jest analiza wykonanych badań porównawczych oraz wskazanie najefektywniejszej metody rozpoznawania choroby.

WYBÓR TEMATU PRACY

- Globalna pandemia COVID-19
 - Wiele przypadków zakażeń w krótkim czasie – potrzeba szybkiej i skutecznej diagnostyki
- Wysoki koszt diagnostyki laboratoryjnej
 - Jedno badanie RT-PCR to koszt 400-500 zł
- Niski koszt diagnostyki radiologicznej
 - Jedno prześwietlenie klatki piersiowej to koszt 20-30 zł
- Dokładność
 - Dużo większa dokładność algorytmów AI, niż pracy radiologów – często o początkowym stanie choroby świadczą niedostrzegalne gołym okiem szczegóły
- Wykształcenie
 - Absolwent PUM - Biotechnologia Medyczna

TEZA PRACY

Analiza porównawcza algorytmów uczenia maszynowego pozwoli na wybranie korzystnego rozwiązania, które usprawni diagnostykę choroby COVID-19.

HIPOTEZA BADAWCZA

Jeżeli zastosujemy uczenie maszynowe
to usprawnimy diagnostykę choroby COVID-19.

ZADANIA W CZĘŚCI TEORETYCZNEJ

- Opis wybranych zagadnień związanych z chorobą COVID-19
- Przedstawienie dostępnego zbioru danych do analizy.
- Omówienie wybranych algorytmów z zakresu uczenia maszynowego oraz ekstrakcji cech z obrazów
- Przedstawienie otrzymanych miar dokładności
- Porównania i wnioski końcowe

ZADANIA W CZĘŚCI PRAKTYCZNEJ

- Przygotowanie zbioru danych do analizy.
- Implementacja algorytmów.
- Przeprowadzenie eksperymentów.

SPIS TREŚCI

Wstęp

1. Cel i zakres pracy

1.1 Środowisko sprzętowe

1.2 Środowisko programistyczne

2 Choroba COVID-19

2.1 Pandemia

2.2 Diagnozowanie

3 Algorytmy klasyfikacji danych

3.1 Klasyfikatory probabilistyczne

3.1.1 Naiwny klasyfikator Bayesa

3.2 Klasyfikatory SVM

3.2.1 Liniowy klasyfikator SVM

3.2.2 Nieliniowy klasyfikator SVM

3.3 Boosting

3.3.1 AdaBoost

3.3.2 GradientBoost

3.4 Sieci konwolucyjne

3.4.1 VGG-19

3.4.2 ResNet-50

3.4.3 DesNet-121

3.4.4 EfficientNet-B0

4 Zbiór danych

4.1 Źródło

4.2 Zawartość

5 Implementacja

6 Wyniki

6.1 Krzywe ROC

6.2 Tabele konfuzji

6.3 Miary jakości

Wnioski

Podsumowanie

Spis literatury

METODY BADAWCZE

- Eksperyment naukowy
 - Wnoszenie zmian do badanych algorytmów, regulowanie ich i kontrolowanie w celu poznania wybranych związków istniejących między przedmiotem badań a otoczeniem.
- Analiza
 - Rozpatrywanie algorytmów pod kątem poprawności oraz szybkości działania.
- Porównanie
 - Ustalenie podobieństw i różnic między badanymi algorytmami.
- Synteza
 - Wyciąganie wniosków na podstawie szczegółowych wyników przeprowadzonych badań.
- Interpretacja
 - Wyjaśnianie otrzymanych wyników.

ZAWARTOŚĆ CZĘŚCI TEORETYCZNEJ

BRAK



ZAWARTOŚĆ CZĘŚCI PRAKTYCZNEJ

ZADANIA W CZĘŚCI PRAKTYCZNEJ

Przygotowanie zbioru danych

Implementacja miar jakości klasyfikacji

Implementacja algorytmów

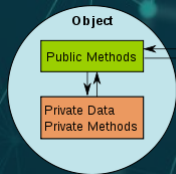
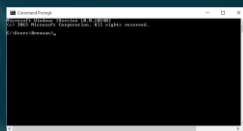
Przeprowadzenie eksperymentu

PROJEKT PROGRAMISTYCZNY



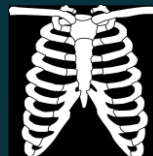
Python 3.10

Aplikacja konsolowa



Programowanie obiektowe

IDE: PyCharm Professional



Dane wejściowe

Algorytmy klasyfikacyjne



Sieci neuronowe

Eksport wyników do



DIAGRAM KLAS

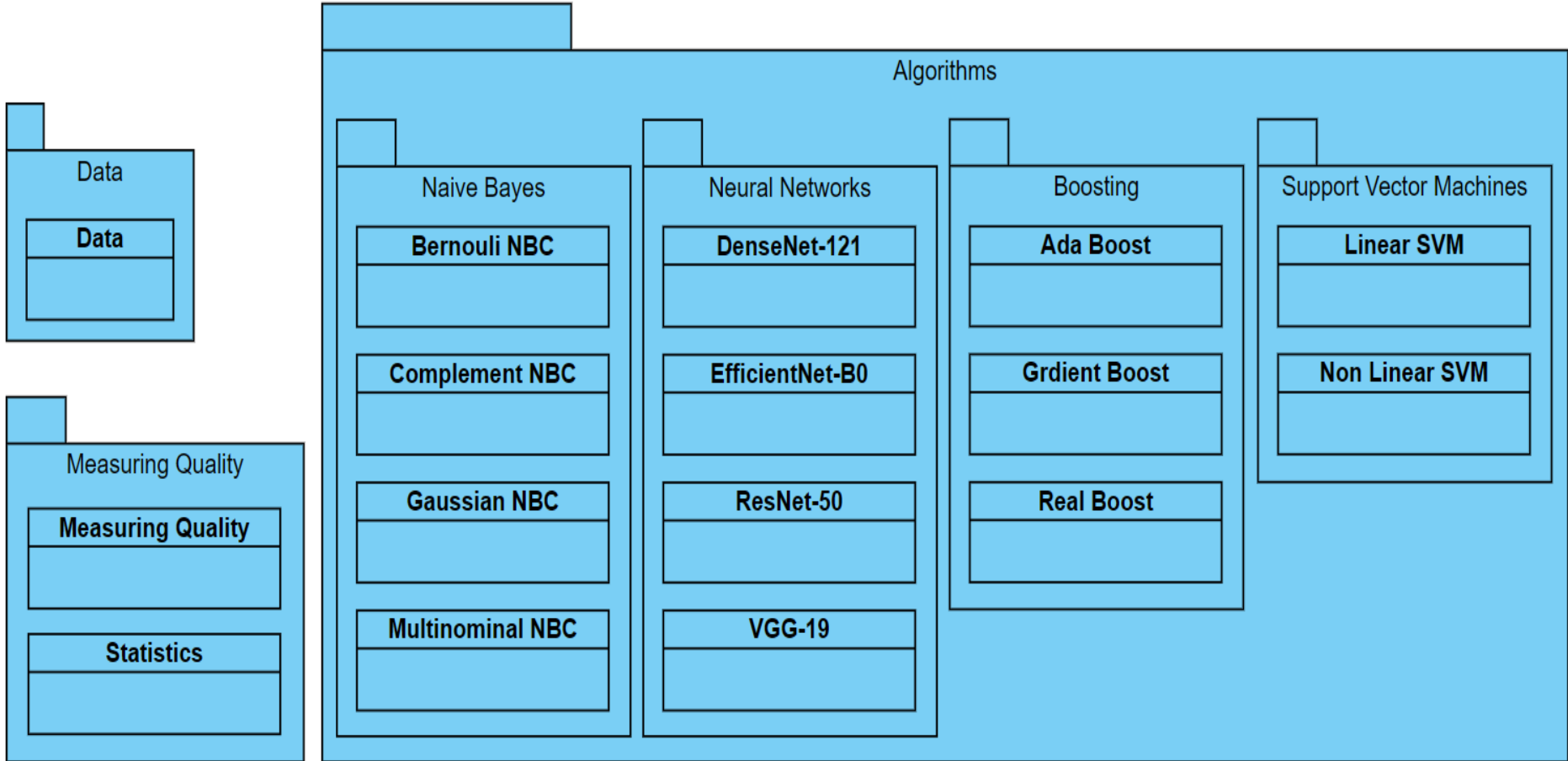
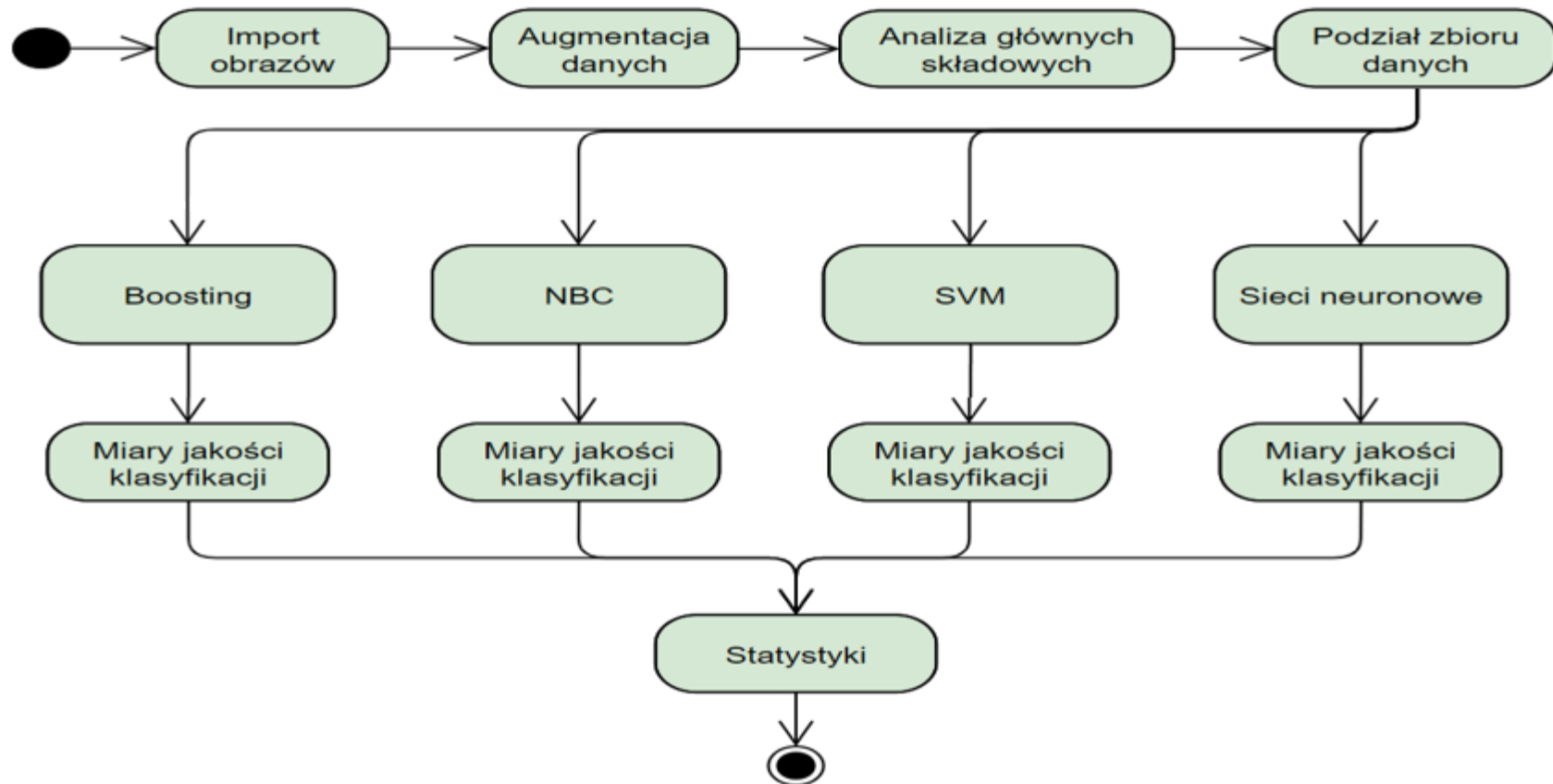


DIAGRAM AKTYWNOŚCI



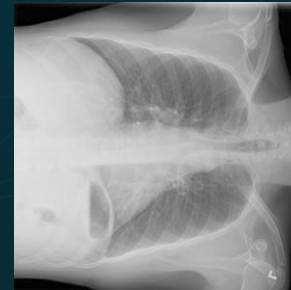
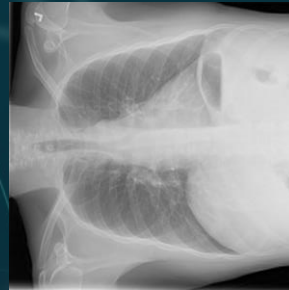
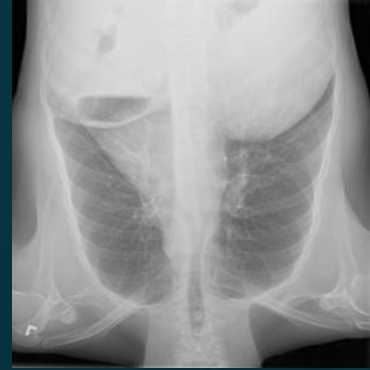
SZCZEGÓŁY IMPLEMENTACJI

DANE WEJŚCIOWE



SZCZEGÓŁY IMPLEMENTACJI

AUGMENTACJA DANYCH



SZCZEGÓŁY IMPLEMENTACJI

ANALIZA GŁÓWNYCH SKŁADOWYCH (PCA)

56	174	53	153	68	32	34	191	22	232	183	32	241	37	250	164	145	13	225
37	76	130	72	120	65	65	64	160	116	2	143	233	32	42	99	190	246	237
229	233	214	251	51	0	182	232	231	92	147	164	100	83	249	229	247	27	111
195	243	65	35	138	55	51	239	44	251	131	121	85	204	250	224	131	216	224
75	61	189	202	107	120	152	155	220	187	26	231	2	185	62	84	134	218	131
132	217	251	213	139	189	95	53	159	94	120	210	106	69	208	41	210	120	104
135	35	133	85	95	126	221	58	93	246	45	238	190	144	203	154	2	95	94
18	174	6	157	243	200	87	232	1	68	172	250	52	145	180	92	217	41	92
87	83	117	186	163	11	53	243	7	160	89	156	176	82	84	21	243	64	211
137	121	204	213	176	179	209	25	80	237	177	121	16	134	244	121	121	234	14
197	177	245	69	146	107	168	102	144	92	132	117	95	218	228	111	67	47	97
208	174	173	255	33	191	9	45	217	0	196	11	178	245	187	172	125	48	73
171	99	106	62	193	240	74	105	90	79	162	4	183	69	62	161	2	109	25
178	9	107	220	153	230	1	218	219	146	219	91	196	145	122	253	92	36	159
191	131	40	250	12	208	129	7	239	214	220	89	24	156	58	22	104	157	80
225	4	126	85	152	210	14	212	42	110	186	56	0	104	189	164	181	229	232
84	53	137	3	48	41	191	140	210	91	103	53	119	243	192	98	132	77	236
76	207	48	41	90	4	54	1	238	37	163	31	81	163	121	163	57	34	228
159	209	109	54	252	84	156	30	248	167	205	127	25	148	15	58	10	170	109
201	31	170	194	103	120	49	160	33	7	141	227	11	82	7	106	67	97	153
224	166	198	240	135	16	76	226	181	240	233	127	108	42	190	40	145	228	0
223	251	136	184	132	108	161	224	115	140	117	219	61	25	44	198	177	99	5
169	130	14	152	78	89	136	160	243	63	35	131	172	168	73	249	44	50	192
149	196	188	11	3	56	106	200	178	54	101	25	211	71	4	120	249	224	240
86	48	217	109	139	253	233	113	13	243	26	228	59	164	206	19	5	128	36
249	104	127	81	72	89	167	188	8	199	23	126	4	27	15	67	58	130	27

1000 x 1000

1 000 000

PCA

100 000

SZCZEGÓŁY IMPLEMENTACJI

PODZIAŁ ZBIORU DANYCH

Podział zbioru danych na dwa podzbiory:

- Uczący
- Testowy

```
def split_data(self, test_size=0.25, random_state=0):  
    """  
    Podział zbioru danych na podzbiory: uczący i testowy.  
    """  
  
    tmp = []  
    for image in self.images:  
        tmp.append(np.asarray(image).flatten())  
    self.images = np.array(tmp)  
    self.X_train, self.X_test, self.y_train, self.y_test = train_test_split(self.images,  
                                     self.labels, test_size=test_size, random_state=random_state)
```

SZCZEGÓŁY IMPLEMENTACJI

ALGORYTMY

```
class DiscreteNBC(BaseEstimator, ClassifierMixin):
    def __init__(self, domain_sizes, laplace=False, logarithm
                 self.laplace_ = laplace
                 self.logarithm = logarithm
                 self.class_labels_ = None

    def fit(self, X, y):
        m, n = X.shape
        self.class_labels_ = np.unique(y)

        y_normalized = np.zeros(m).astype("int32")
        for yy, label in enumerate(self.class_labels_):
            indexes = y == label
            y_normalized[indexes] = yy

        self.PY_ = np.zeros(self.class_labels_.size)
        for yy, label in enumerate(self.class_labels_):
            self.PY_[yy] = (y == label).sum() / m

        self.P_ = np.empty((self.class_labels_.size, n), dtype=object)
        for yy, label in enumerate(self.class_labels_):
            for j in range(n):
                self.P_[yy, j] = np.zeros(self.domain_sizes[j])
```

```
    def predict_proba(self, X):
        m, n = X.shape
        if self.logarithm:
            probs = np.zeros((m, self.class_labels_.size))
        else:
            probs = np.ones((m, self.class_labels_.size))

        for i in range(m):
            x = X[i]
            for y in range(self.class_labels_.size):
                if self.logarithm:
                    for j in range(n):
                        probs[i, y] += self.P_[y, j][x[i, j]]
                    probs[i, y] += self.PY_[y]
                    #probs[i] = np.exp(probs[i])
                else:
                    for j in range(n):
                        probs[i, y] *= self.P_[y, j][x[i, j]]
                    probs[i, y] *= self.PY_[y]

            s = probs[i].sum()
            if s > 0:
                probs[i] /= s

        return probs
```

SZCZEGÓŁY IMPLEMENTACJI

ALGORYTMY

```
class DNBC:
    def __init__(self, data):
        self.data = data
        self.classifier = DiscreteNBC()

    def start(self):
        train_time_start = time()
        self.classifier.fit(self.data.X_train, self.data.y_train)
        train_time_stop = time()

        predict_time_start = time()
        y_pred = self.classifier.predict(self.data.X_test)
        y_score = self.classifier.predict_proba(self.data.X_test)
        predict_time_stop = time()

        return MeasuringQuality("DNBC",
                                "Discrete Naive Bayes classifier",
                                train_time_stop - train_time_start,
                                predict_time_stop - predict_time_start, self.data.y_test, y_pred, y_score)
```


WYNIKI DZIAŁANIA PROGRAMU

MIARY JAKOŚCI KLASYFIKACJI



ALGORYTM

KLASA

MACIERZE KONFUZJI

MIARY JAKOŚCI KLASYFIKACJI

CZUŁOŚĆ

SPECYFICZNOŚĆ

DOKŁADNOŚĆ

BŁĄD

F1

KRZYWE ROC

WYNIKI DZIAŁANIA PROGRAMU

PRZYKŁAD - NBC

Dane wejściowe: 1000 -> 2200

Podział: 1650 / 550

Czas uczenia: 6.570717

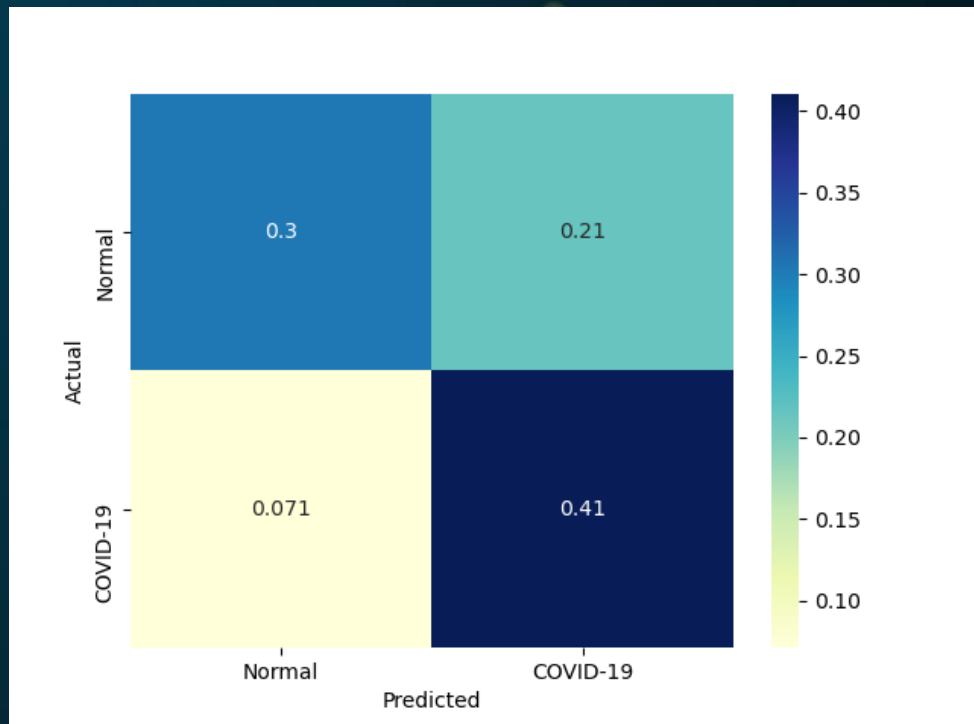
Czas predykcji: 2.751148

Wyniki prawdziwie pozytywne: 226

Wyniki prawdziwie negatywne: 167

Wyniki fałszywie pozytywne: 118

Wyniki fałszywie negatywne: 39



WYNIKI DZIAŁANIA PROGRAMU

PRZYKŁAD - NBC

Czułość: 0.852830

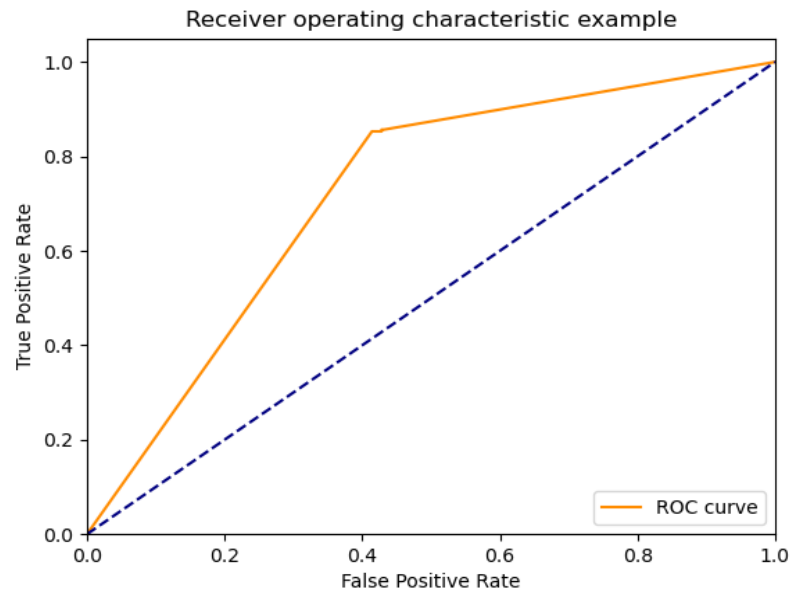
Specyficzność: 0.585965

Precyzja: 0.656977

Dokładność: 0.714545

Błąd: 0.285455

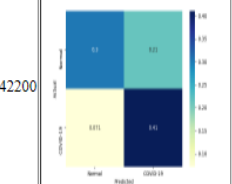
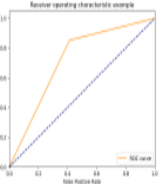
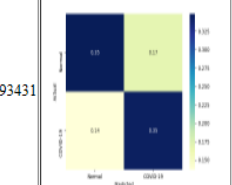
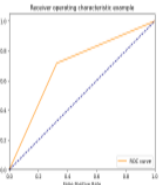
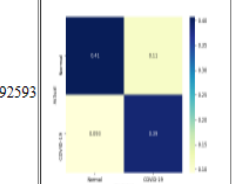
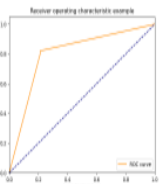
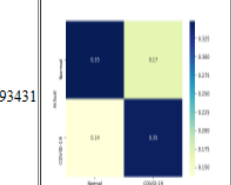
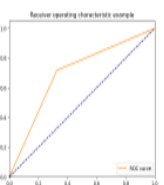
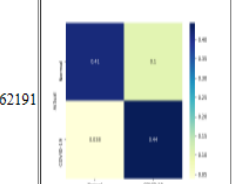
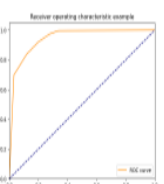
F1: 0.742200



WYNIKI DZIAŁANIA PROGRAMU

Dla każdego z badanych algorytmów wynikiem działania programu jest zestaw takich danych.

Zestawienie wszystkich tych danych będzie podstawą późniejszej analizy w części teoretycznej.

	method	description	train_time	predict_time	true_positive	true_negative	false_positive	false_negative	sensitivity	specificity	precision	accuracy	error	f1	confusion_matrix	roc_curve
0	BernoulliNBC	Naive Bayes classifier for multivariate Bernoulli models	6.570717	2.751148	226	167	118	39	0.852830	0.585965	0.656977	0.714545	0.285455	0.742200		
1	ComplementNBC	The Complement Naive Bayes classifier described in Rennie et al	4.844363	1.477513	190	192	93	75	0.716981	0.673684	0.671378	0.694545	0.305455	0.693431		
2	GaussianNBC	Gaussian Naive Bayes (GaussianNB)	9.401089	10.515112	214	224	61	51	0.807547	0.785965	0.778182	0.796364	0.203636	0.792593		
3	MultinomialNBC	Naive Bayes classifier for multinomial models	5.881502	1.615433	190	192	93	75	0.716981	0.673684	0.671378	0.694545	0.305455	0.693431		
4	KNeighborsClassifier	Classifier implementing the k-nearest neighbors vote.	0.000750	24.782750	244	228	57	21	0.920755	0.800000	0.810631	0.858182	0.141818	0.862191		

I SPOSÓB WERYFIKACJI POPRAWNOŚCI



WYNIKI TESTOWANIA

test_ada_boost 8.39 s

■ **TestAdaBoost** 8.39 s

■ **test_start** **passed** 8.39 s

test_bernoulli_nbc 1.86 s

■ **TestBernoulliNBC** 1.86 s

■ **test_start** **passed** 1.86 s

test_gaussian_nbc 2.01 s

■ **TestGaussianNBC** 2.01 s

■ **test_start** **passed** 2.01 s

test_gradient_boost 6.16 s

■ **TestGradientBoost** 6.16 s

■ **test_start** **passed** 6.16 s

test_linear_svm 1.59 s

■ **TestLinearSVM** 1.59 s

■ **test_start** **passed** 1.59 s

test_nonlinear_svm 2.66 s

■ **TestNonLinearSVM** 2.66 s

■ **test_start** **passed** 2.66 s

II SPOSÓB WERYFIKACJI POPRAWNOŚCI

Testy jednostkowe komponentów:

- Odczyt danych z plików wejściowych
- Przygotowanie zbioru danych do analizy
- Obliczanie miar jakości klasyfikacji
- Przygotowanie statystyk (wyników eksperymentu)

WYNIKI TESTOWANIA

test_data 16.07 s

■ **TestData** 16.07 s

test_augment_data passed 3.70 s

test_dump_data passed 1.85 s

test_import_data passed 2.96 s

test_load_data passed 1.75 s

test_resize_data passed 2.85 s

test_split_data passed 2.94 s

test_statistics 3.07 s

■ **TestStatistics** 3.07 s

test_create_statistics passed 1.25 s

test_export_csv passed 255 ms

test_export_html passed 362 ms

test_insert passed 122 ms

test_show passed 221 ms

test_update_data passed 855 ms

test_measuring_quality 924 ms

■ **TestMeasuringQuality** 924 ms

test_calculate passed 924 ms

WYMAGANE ZASOBY KOMPUTEROWE

CPU: Dowolny, wielordzeniowy

GPU: Karta GPU NVIDIA® z architekturą CUDA® 3.5, 5.0, 6.0, 7.0, 7.5, 8.0 i nowszą niż 8.0.

PAMIĘĆ: ok. 4GB

RAM: min. 64GB

OS: MS Windows 8, 10, 11 (x64)

Wymagania Systemowe:

- Sterowniki NVIDIA GPU -CUDA® 11,2 wymaga 450.80.02 lub wyższej.
- CUDA® Toolkit -TensorFlow obsługuje CUDA® 11,2 (TensorFlow> = 2.5.0)
- CUPTI statki z CUDA® Toolkit.
- cuDNN SDK 8.1.0 wersje cuDNN).

TRUDNOŚCI W REALIZACJI PRACY

SPRZĘT

Brak dostępu do maszyny o dużej wydajności.

Czas wykonania połowy algorytmów na 10% danych
aktualnie zajmuje około 8 godzin.

POSTĘPY W REALIZACJI

- ✓ Przygotowanie zbioru danych
- ✓ Przygotowanie miar jakości klasyfikacji
- ✓ Przygotowanie statystyk z eksperymentu
- ✓ Naiwny klasyfikator Bayesa
- ✓ Liniowy klasyfikator SVM
- ✓ Nieliniowy klasyfikator SVM
- ✓ AdaBoost
- ✓ GradientBoost
- ✗ RealBoost
- ✗ VGG-19
- ✗ ResNet-50
- ✗ DesNet-121
- ✗ EfficientNet-B0

62 %

✗ Część teoretyczna

0 %

HARMONOGRAM REALIZACJI

Implementacja
algorytmów NBC, SVM, RB

październik

listopad

Implementacja sieci
neuronowych

Przeprowadzenie badań,
analiza i opracowanie
wyników i wniosków

grudzień

styczeń

Część opisowa

BIBLIOGRAFIA

- Dokumentacja biblioteki Scikit-learn
- Materiały własne promotora
- „Techniki szybkiej detekcji: ekstrakcja cech poprzez obrazy całkowe, boosting, kaskady klasyfikatorów” Przemysław Klęsk
- Materiały dydaktyczne z przedmiotu Sztuczna inteligencja (wykłady, notatki, programy z laboratoriów)



PYTANIA

DZIĘKUJĘ ZA UWAGĘ!

Postępy, wyniki, szczegóły dostępne pod adresem:
wl44545.github.io

Kontakt:
lukasz_wieckowski@zut.edu.pl

CREDITS: This presentation template was created by **Slidesgo**, including icons by **Flaticon**, and infographics & images by **Freepik**.