Analityka i eksploracja danych - Data Mining

prowadzący: dr hab. inż. Henryk Maciejewski

Jarosław Ciołek-Żelechowski(*218386*)

31.01.2020

1 Cel projektu

Celem projektu było zaproponowanie modelu do predykcji danych dotyczących spamu mailowego. Dostarczone dane miały reprezentowały zawartość 4789 maili. Na jeden mail składało się 463 kolumny, opisujące obecność lub brak konkretnego tag-u.

	ACT_NOW	ADDRESSES_ON_CD	ADULT_SITE	ADVERT_CODE	ADVERT_CODE2	ALL_CAPS_HEADER	ALL_CAP_PORN	ALL_NATURAL	AMATEUR_PORN
0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0

5 rows × 463 columns

Rysunek 1: Prezentacja danych

1.1 Sposób oceny modeli

W celu określenia jakości otrzymanego wyniku, skorzystałem z szeregu statystyk:

- **accuracy** - procent poprawnych klasyfikacji. Stosunek ilości poprawnych predykcji dla danej klasyfikacji, opisana wzorem:

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

  , gdzie y to wartość prawdziwa, a $\hat{y}$ to wartość naszej predykcji,

- **precision** - umiejętność klasyfikatora do niepoprawnej klasyfikacji próbek negatywnych jako pozytywnych, najprościej opisać wzorem:

$$\texttt{precision} = \frac{tp}{(tp + fp)}$$

  , gdzie tp to ilość prawdziwie pozytywnych predykcji(poprawnie sklasyfikowane *prawdy*), a fp to ilość fałszywie pozytywnych(niepoprawnie sklasyfikowanych *prawd*),

- **recall** - umiejętność klasyfikatora do poprawnej klasyfikacji, opisana wzorem:
$$\texttt{recall} = \frac{tp}{(tp + fn)}$$
, gdzie tp to ilość prawdziwie pozytywnych predykcji(poprawnie sklasyfikowane *prawdy*), a fn to ilość fałszywie negatywnych(niepoprawnie sklasyfikowanego *fałszu*),

- **fscore** - miara bardziej skomplikowana, będąca średnią ważoną dwóch powyższych,
$$F1 = \frac{2 * (precision * recall)}{(precision + recall)}$$

- **Confusion Matrix** - tablica błędów. Graficzna reprezentacja wszystkich wykonanych predykcji w postaci macierzy stanu faktycznego i wyliczonego przez nasz klasyfikator. Dla idealnej klasyfikacji zaznaczona jest tylko przekątna takiej macierzy.

W celu rozróżnieniu dwóch wyników w trakcie eksperymentu porównywałem jego **fscore** i wybieraliśmy ten z większym wynikiem. W dalszej części projektu jednak dużo większą uwagę kładłem na poprawianie wartości z tablicy błędów.

1.2 Walidacja krzyżowa

Trenowanie i testowanie klasyfikatorów opisanych w tej pracy odbyło się z wykorzystaniem 5x2cv, czyli 5-krotnej walidacji krzyżowej. Oznacza to że 5 razy losowo podzieliliśmy zbiór danych na dane uczące i testujące, wykonaliśmy badania i zamieniliśmy zbiory uczące i testujące miejscami.

2 Przegląd modeli

W pierwszym etapie prac, chciałem korzystając z domyślnych klasyfikatorów, sprawdzić jak sobie one poradzą z danymi. Tym samym chciałem sprawdzić w który z klasyfikatorów warto zainwestować więcej czasu. Uznałem że rozpatrzę następujące modele:

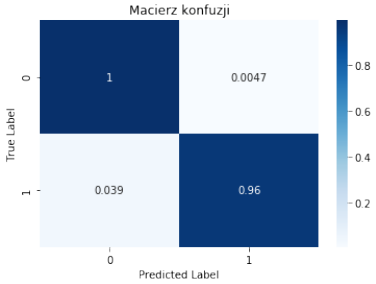
- Maszyna wektorów nośnych,
- K-najbliższych sąsiadów,
- Drzewo decyzyjne,
- Naiwnego Beyesa,
- Sieć Neuronową.

2.1 Maszyna wektorów nośnych

Accuracy	Precision	Recall	F-score
0.9834	0.9836	0.9834	0.9834

Macierz konfuzji:

```
[[2962   14]
 [  70 1744]]
```



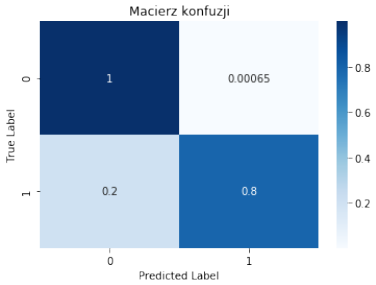
Rysunek 2: Maszyna wektorów nośnych

2.2 K-najbliższych sąsiadów

Accuracy	Precision	Recall	F-score
0.9173	0.9264	0.9173	0.9150

Macierz konfuzji:

```
[[3078    2]
 [ 350 1360]]
```



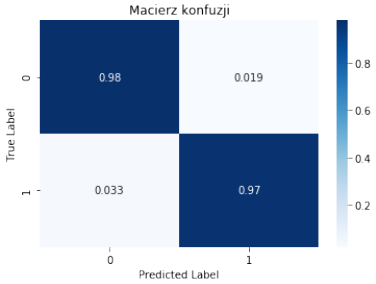
Rysunek 3: K-najbliższych sąsiadów

2.3 Drzewo decyzyjne

Accuracy	Precision	Recall	F-score
0.9741	0.9741	0.9741	0.9741

Macierz konfuzji:

```
[[2858   56]
 [  62 1814]]
```



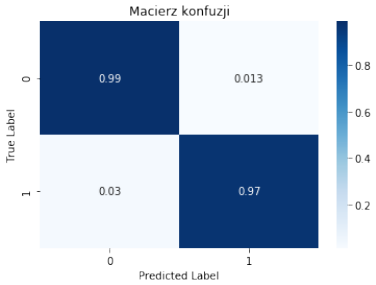
Rysunek 4: Drzewo decyzyjne

2.4 Naiwny Bayes

Accuracy	Precision	Recall	F-score
0.9794	0.9794	0.9794	0.9794

Macierz konfuzji:

```
[[2962   38]
 [  54 1736]]
```



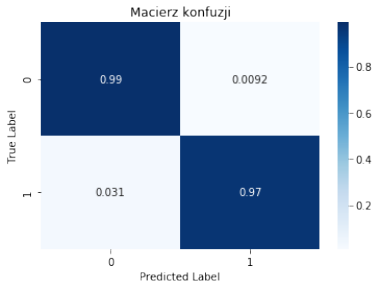
Rysunek 5: Naiwny Bayes

2.5 Sieć Neuronowa

Accuracy	Precision	Recall	F-score
0.9795	0.9796	0.9795	0.9795

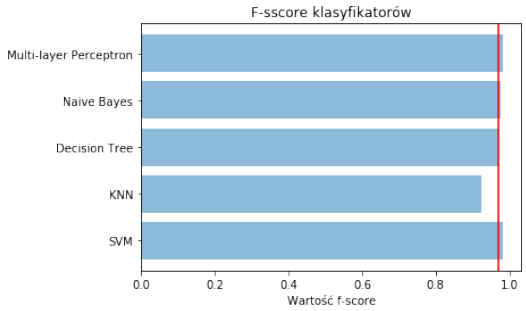
Macierz konfuzji:

```
[[2911   27]
 [  58 1794]]
```



Rysunek 6: Sieć Neuronowa

2.6 Wybór najlepszych



Rysunek 7: Wybór najlepszych

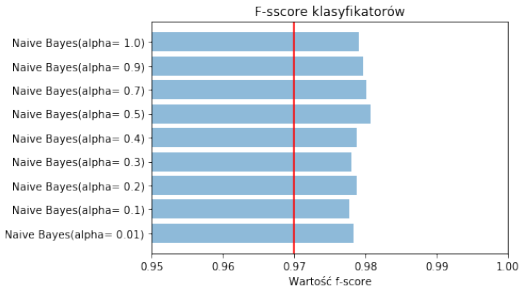
, czyli do następnego *etapu* przechodzą:

- Naiwny Beyes,
- Maszyna wektorów nośnych,
- Sieć Neuronowa.

3 Dostrajanie hiper-parametrów

3.1 Naiwny Bayes

Przy tym modelu, mamy tak na prawdę dwa hiperparametry, którymi możemy sterować - *alpha* i *fit_prior*. Pierwszy odpowiada on za parametr wygładzania i przyjmuje wartości z zakresu od 0 do 1. Drugi dotyczy prawdopodobieństw klas i może być prawdziwy lub nie.



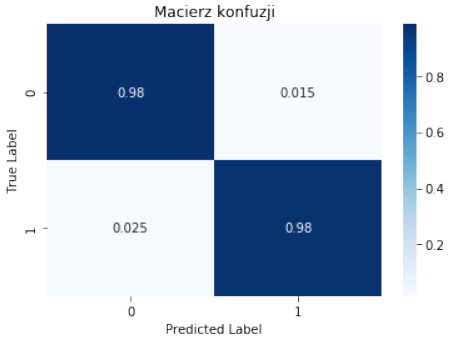
Rysunek 8: Zależność **alphy**

Jak widać na powyższym rysunku, różnice w fscoreach są niewielkie ale najlepszy wyniki otrzymuje się dla **alphy=0.5**

Accuracy	Precision	Recall	F-score
0.9786	0.9786	0.9786	0.9786

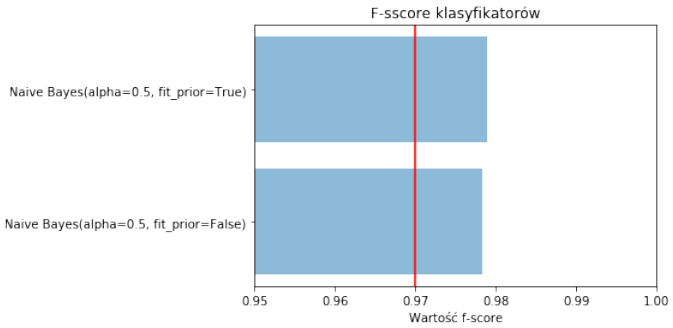
Macierz konfuzji:

```
[[2956   46]
 [  44 1744]]
```



Rysunek 9: Naiwny Beyes(alpha=0.5)

Jak to wygląda dla drugiego parametru?



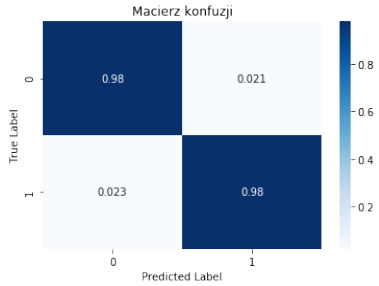
Rysunek 10: Naiwny Beyes(alpha=0.5)

, lepiej dla **fit_prior** ustawionego na **False**.

Accuracy	Precision	Recall	F-score
0.9786	0.9787	0.9786	0.9786

Macierz konfuzji:

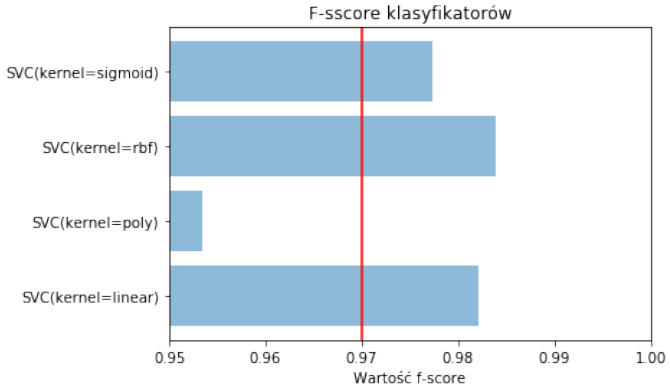
```
[[2862   62]
 [  42 1824]]
```



Rysunek 11: Naiwny Beyes(alpha=0.5, fit_prior=False)

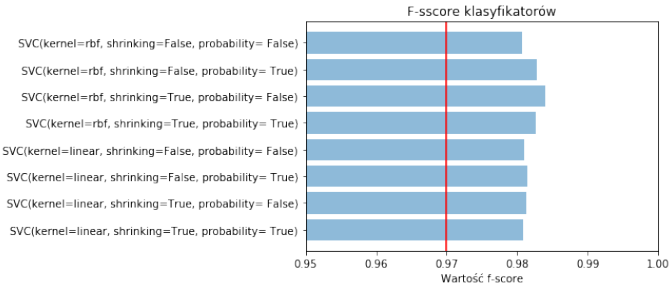
3.2 Maszyna wektorów nośnych

Dla SVM-u najważniejszym hiper-parametrem jest *kernel* i to od jego wyboru chciałbym zacząć.



Rysunek 12: klasyfikacja SVMów w zależności od kernelu

Jak widać najlepsze wyniki uzyskuje dla **'rbf'** i **'linear'**. Kolejnymi hiperparametrami, którymi się zająłem było *shrnking* i *probability*. Oba parametry są zmiennymi typu bool. Pierwszy z nich pozwala na używanie zmniejszającej heurystyki, podczas gdy drugi pozwala na estymację prawdopodobieństwa.



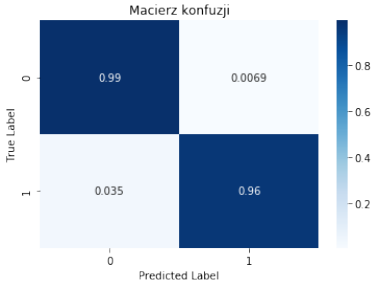
Rysunek 13: klasyfikacja SVMów w zależności od shrnking i probability

Najlepszy wynik uzyskałem dla **shrnking=True** i **probability=False**.

Accuracy	Precision	Recall	F-score
0.9819	0.9821	0.9819	0.9819

Macierz konfuzji:

```
[[2890   20]
 [  66 1814]]
```

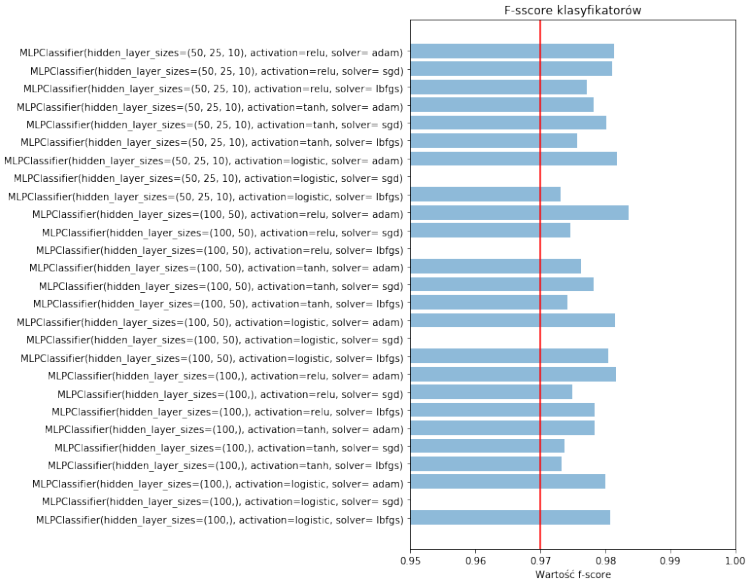


Rysunek 14: SVC(kernel=rbf, shrinking=True, probability= False)

3.3 Sieć Neuronowa

Dla Sieci Neuronowej zmiennych jest dużo więcej:

- hidden_layer_sizes - czyli budowa samej sieci,
- activation - czyli funkcja aktywacji,
- solver - czyli optimizer dla sieci



Rysunek 15: Przegląd klasyfikatorów MLP

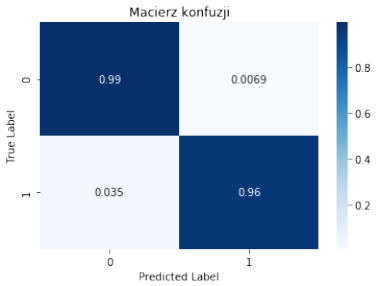
Jak widać na wykresie powyżej najlepszy wynik otrzymała sieć neuronowa dla następujących parametrów:

- **hidden_layer_sizes=(100,)**,
- **activation=relu**,
- **solver= adam**.

Accuracy	Precision	Recall	F-score
0.9837	0.9837	0.9837	0.9837

Macierz konfuzji:

```
[[2933   25]
 [  55 1777]]
```



Rysunek 16: MLPClassifier(hidden_layer_sizes=(100,), activation=relu, solver= adam)

3.4 Wybór najlepszych

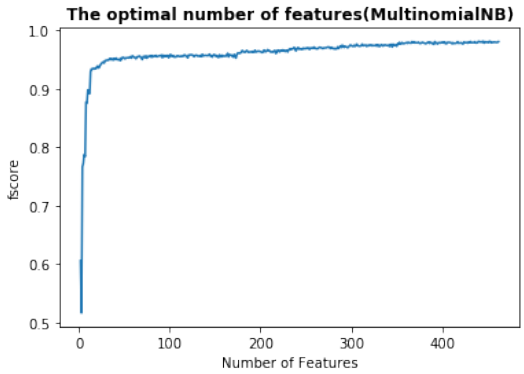
Podsumowując do tej pory najlepsze modele to:

- MultinomialNB(alpha=0.5, fit_prior=True),
- svm.SVC(kernel='rbf', gamma='scale', shrinking=True, probability= False),
- MLPClassifier(hidden_layer_sizes=(100,50,), activation='relu', solver='adam').

4 Ranking cech

Mając już wybrane modele można się zastanowić jak wpłynie na nie zmiana wymiarowości danych. Postanowiłem dla każdego najlepszego modelu, zrobić ranking cech w oparciu o test statystyczny chi-kwadrat.

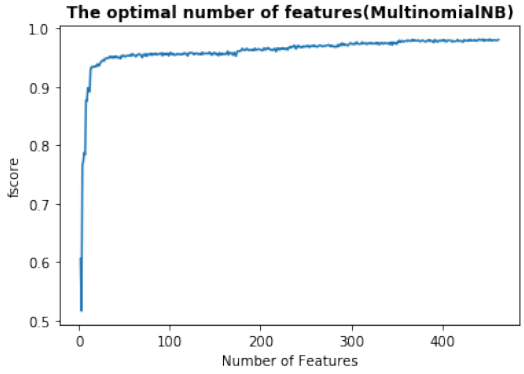
4.1 Naiwny Bayes



Rysunek 17: Wpływ liczby kolumn na wynik

Najlepszy wynik uzyskano dla: 453 z 463

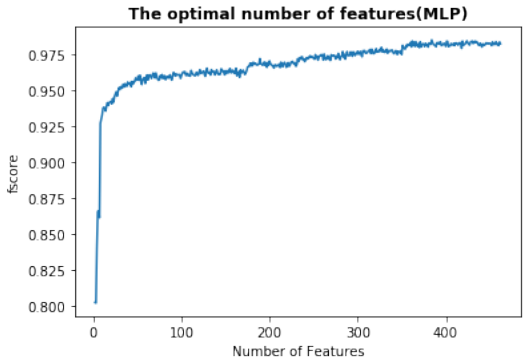
4.2 SVM



Rysunek 18: Wpływ liczby kolumn na wynik

Najlepszy wynik uzyskano dla: 367 z 463

4.3 Multilayer Perceptron



Rysunek 19: Wpływ liczby kolumn na wynik

Najlepszy wynik uzyskano dla: 385 z 463

4.4 Wniosek

Jak widać dla wszystkich modeli dość szybko dodając dane zaczniemy uzyskiwać dość dobre rezultaty. Wystarczy ok. 50 elementów(czyli redukcja cech o prawie 90%!), by zacząć uzyskiwać wyniki w okolicach .90 fscora.

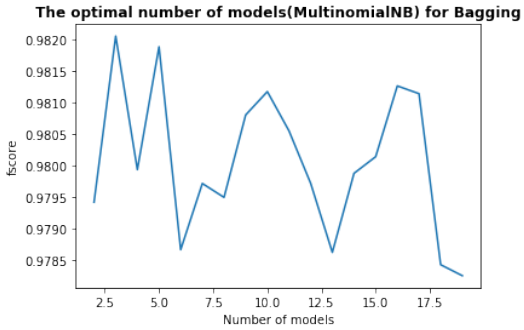
5 Bagging

Skorzystałem również z możliwości przeprowadzenia Baggingu, czyli złożenia n klasyfikatorów. Polega on na tym, że pomiędzy n konkretnych klasyfikatorów zostaje losowo podzielony zbiór danych. Następnie każdy z n klasyfikatorów jest uczony. Gdy przychodzi do takiego baggingowego klasyfikatora zapytanie predict, powiela je on na n klasyfikatorów na które się składa, i następnie pozwala im na głosownie.

W moim przypadku zbudowałem klasyfikator baggingowy dla każdego z klasyfikatór o których psiałem wcześniej, budując go o dane dla których działa najlepiej.

Jako że klasyfiktor baggingowy przyjmuje ilość klasyfikatorów jako zmienną, postanowiłem sprawdzić jaka ilość z przedziału od 2 do 20 będzie najlepsza.

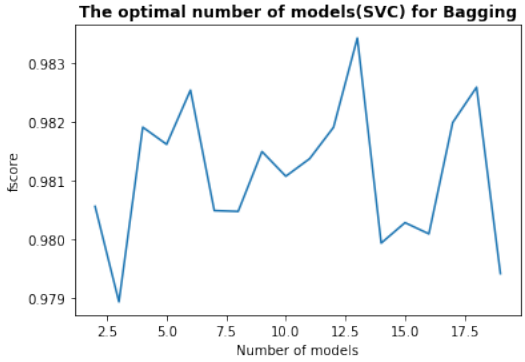
5.1 Naiwny Bayes



Rysunek 20: Wpływ ilości klsyfikatorów na Bagging dla NBa

Najlepsza ilość: **3**

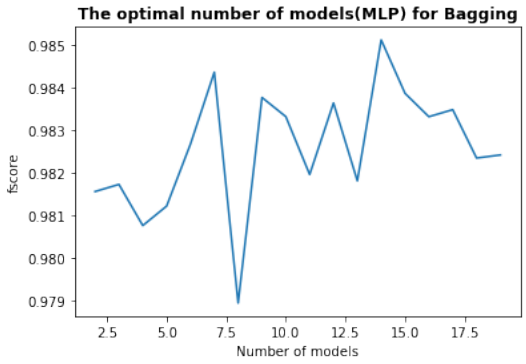
5.2 SVM



Rysunek 21: Wpływ ilości klsyfikatorów na Bagging dla SVMa

Najlepsza ilość: **13**

5.3 Multilayer Perceptron



Rysunek 22: Wpływ ilości klsyfikatorów na Bagging dla MLPa

Najlepsza ilość: **14**

6 Prezentacja najlepszych wyników

Poniżej przedstawiam najlepsze modele jakie udało mi się uzyskać dla poszczególnych klasyfikatorów:

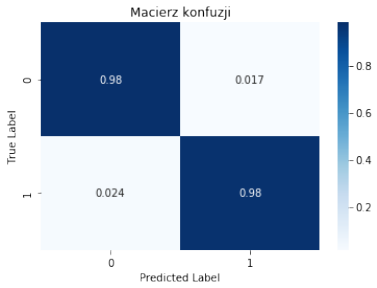
6.1 Naiwny Beyes

Klasyfikator MultinomialNB(alpha=0.5, fit_prior=True)

Accuracy	Precision	Recall	F-score
0.9812	0.9812	0.9812	0.9812

Macierz konfuzji:

```
[[2844   48]
 [  46 1852]]
```



Rysunek 23: MultinomialNB(alpha=0.5, fit_prior=True)

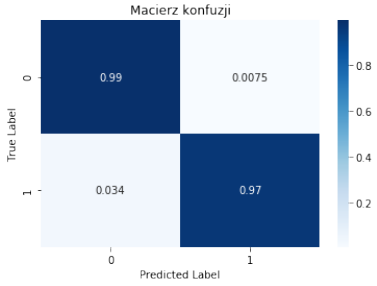
6.2 Maszyna wektorów nośnych

Klasyfikator SVC(kernel=rbf, gamma=scale, shrinking=True, probability= False)

Accuracy	Precision	Recall	F-score
0.9841	0.9842	0.9841	0.9840

Macierz konfuzji:

```
[[2910   22]
 [  64 1794]]
```



Rysunek 24: SVC(kernel=rbf, gamma=scale, shrinking=True, probability= False)

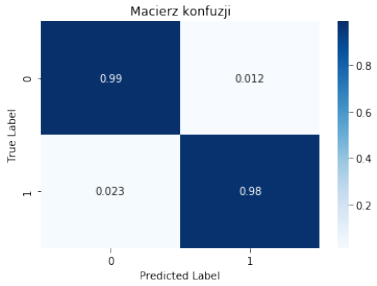
6.3 Sieć Neuronowa - najlepszy klasyfikator

Bagging, 14 klsyfikatorów MLP(hidden_layer_sizes=(100,50,), activation=relu, solver=adam)

Accuracy	Precision	Recall	F-score
0.9833	0.9833	0.9833	0.9833

Macierz konfuzji:

```
[[2937   35]
 [  41 1777]]
```



Rysunek 25: Bagging_14_MLPClassifier(hidden_layer_sizes=(100,50,), activation=relu, solver=adam)

7 Wnioski

Wydaje mi się że najważniejszym wnioskiem jaki można było wynieść z tego projektu jest to że samo znalezienie modelu, który pasuje do naszych danych nie jest jeszcze sukcesem. Cała praca, i cały ciężar zadania spoczywa na dostrajaniu hiper-parametrów i podejmowania w oparciu o to coraz lepszych decyzji.