METODY ILOŚCIOWE W BADANIACH EKONOMICZNYCH
Tom XXII/1, 2021, s. 29 – 38

ZASTOSOWANIE BIBLIOTEKI ML.NET DO BADAŃ EKONOMICZNYCH

Waldemar Karwowski https://orcid.org/0000-0002-9988-0209
Instytut Informatyki Technicznej
Szkoła Główna Gospodarstwa Wiejskiego w Warszawie
e-mail: waldemar_karwowski@sggw.edu.pl

Streszczenie: W artykule podjęto próbę oceny przydatności biblioteki ML.NET do badań w dziedzinie ekonomii. Przedstawiono możliwości wykorzystania uczenia maszynowego w aplikacjach i krótko omówiono dostępne bezpłatnie biblioteki programistyczne związane z uczeniem maszynowym. Omówiono funkcjonalność biblioteki ML.NET ze szczególnym uwzględnieniem przydatności do badań i zastosowań w dziedzinie ekonomii. W celu weryfikacji możliwości ML.NET do generacji modelu jak i wykorzystania go do prognozy, utworzono prostą aplikację w języku C# prognozującą poziom inflacji na podstawie zgromadzonych danych.

Słowa kluczowe: uczenie maszynowe, ML.NET, zastosowanie informatyki w ekonomii

JEL classification: C80

WSTĘP

W ostatnich latach na całym świecie bardzo popularne stało się wykorzystanie technik uczenia maszynowego (ang. machine learning). Techniki te są integralną częścią sztucznej inteligencji najogólniej rozumianej jako zdolność systemów informatycznych do rozwiązywania zadań, które gdy wykonywane są przez człowieka wymagają inteligencji. W szczególności dotyczy to zadań, w których interpretujemy nowe dane na podstawie wcześniej zgromadzonej wiedzy, na ogół wiedza ta zawarta jest w przygotowanym modelu. Uczenie maszynowe bazuje głównie na metodach matematycznych, przede wszystkim metodach statystycznych oraz nowoczesnych narzędziach przetwarzania informacji

https://doi.org/10.22630/MIBE.2021.22.1.3

takich jak sieci neuronowe. Bardzo często przygotowanie danych w celu utworzenia modelu wymaga współpracy człowieka, na przykład oznaczenia przypadków jako pozytywne lub negatywne. Rozwój i szerokie zastosowanie uczenia maszynowego w dużym stopniu stały się możliwe dzięki wzrastającej sile obliczeniowej komputerów. Do realizacji zadań uczenia maszynowego konieczne są nie tylko wydajne komputery ale przede wszystkim specjalistyczne oprogramowanie. Powstało wiele narzędzi oraz bibliotek pomagających zarówno w tworzeniu modeli jak i korzystaniu z nich. Gotowe narzędzia przeznaczone są zasadniczo dla wszystkich użytkowników, oparte są one przede wszystkim na interfejsie graficznym. Biblioteki przeznaczone są głównie dla programistów, ułatwiają tworzenie zintegrowanych rozwiązań, które oprócz innej funkcjonalności umożliwiają tworzenie, trenowanie i wykorzystanie modeli. Obecnie, ze względu na dużą popularność uczenia maszynowego, zwiększa się liczba dostępnych rozwiązań przeznaczonych dla różnych systemów operacyjnych i różnych języków programowania. W dużym stopniu są to biblioteki możliwe do bezpłatnego wykorzystania, często z otwartym kodem źródłowym. Najbardziej znane aplikacje komercyjne to IBM Watson Studio, IBM SPSS Modeler, Mathematica, MATLAB, SAS Enterprise Miner czy Oracle Data Mining; oferują one możliwość pracy z interfejsem graficznym oraz udostępniają interfejs programistyczny. Obecnie wiele aplikacji jest udostępniane jako usługi w chmurze, najbardziej znane to Amazon Machine Learning, Azure Machine Learning czy Oracle AI Platform Cloud Service. Wśród aplikacji bezpłatnych są zarówno aplikacje z interfejsem graficznym takie jak Deep Learning Studio, które nie wymagają nic więcej niż wklejenie tabeli z danymi i użycia myszy komputerowej. Przede wszystkim są to jednak biblioteki programistyczne. Obecnie najpopularniejszym językiem programowania w dziedzinie sztucznej inteligencji jest Python i najczęściej używane biblioteki z implementacją algorytmów uczenia maszynowego są napisane w Pythonie, który umożliwia pracę interaktywną i jednocześnie dostarcza dodatkowe biblioteki np. do obliczeń matematycznych takie jak NumPy, SciPy, oraz do obróbki i wizualizacji danych takie jak Pandas i Matplotlib. Należy jednak zaznaczyć, że krytyczne fragmenty oprogramowania są w rzeczywistości napisane w języku C/C++ a tylko interfejs użytkownika jest napisany w Pythonie. Najpopularniejsze biblioteki do uczenia maszynowego to: TensorFlow i PyTorch, ponadto Keras (wykorzystuje TensorFlow) oraz SciKit Learn (bazuje na NumPy i SciPy) [Gevorkyan i in. 2019]. Ponieważ Python jest językiem skryptowym, to bardzo dobrze nadaje się do pracy interaktywnej natomiast nieco gorzej wymienione biblioteki integrują się ze środowiskami programistycznymi takimi jak Java czy .NET. Z jednej strony tworzone są interfejsy programistyczne w Javie czy C# do wymienionych wyżej bibliotek z drugiej strony tworzone są dedykowane biblioteki dla danego języka programowania. W Javie przykładem takiego rozwiązania jest Weka czy Java-ML, natomiast w środowisku .NET jest to ML.NET. ML.NET jest biblioteką dostępną bezpłatnie, rozwijaną bezpośrednio przez Microsoft od roku 2018 [Zeeshan i in. 2019; Capellman 2020]. Pierwsza

pełna wersja 1.0 została wydana w połowie roku 2019 i jest systematycznie rozwijana, obecnie dostępna jest wersja 1.7.

Celem artykułu jest zbadanie przydatności biblioteki ML.NET do zagadnień w dziedzinie ekonomii. Platforma .NET i język C# należą do jednych z najpopularniejszych środowisk programistycznych dla aplikacji pracujących w systemie Windows. Aplikacje te często dotyczą zagadnień związanych z ekonomią. Możliwość łatwego rozszerzenia takich aplikacji o możliwości uczenia maszynowego stwarza nową jakość i pozwala spełnić wymagania współczesnych użytkowników.

MOŻLIWOŚCI ML.NET

ML.NET powstał z potrzeby umożliwienia programistom .NET integracji aplikacji z algorytmami uczenia maszynowego. ML.NET daje możliwość dodawania uczenia maszynowego do aplikacji .NET w scenariuszach online lub offline. Microsoft już wcześniej udostępniał usługi związane z uczeniem maszynowym na komercyjnej platformie Microsoft Azure, natomiast ML.NET jest narzędziem bezpłatnym o otwartym kodzie źródłowym. Jednym z jego założeń jest unifikacja procesów rozwojowych modeli oraz aplikacji, w których maja one być wykorzystane. Biblioteka uporządkowana jest w formie przestrzeni nazw: główna przestrzeń ML zawiera podstawowe klasy i metody rozszerzające, ML.Data zawiera komponenty do ładowania i zapisu danych, definicji schematów danych oraz metryk dla modeli; ML.Transforms zawiera komponenty do transformacji danych; ML.Trainers zawiera trenery, parametry modeli i narzędzia. Dodatkowo w ML.Calibrators zawarte są metody, które obliczają prawdopodobieństwa z wyników dla klasyfikatorów binarnych. Schemat korzystania z biblioteki jest następujący: zbieranie, ładowanie i przekształcanie danych treningowych, określanie potoku operacji przede wszystkim algorytmu uczenia maszynowego, trenowania modelu, ocena modelu, zapisanie modelu do wykorzystania w aplikacji do tworzenia prognoz. Po etapie oceny modelu jest możliwa korekta danych i ponowne trenowanie modelu. Najważniejszym elementem w ML.NET jest model uczenia maszynowego. Model określa kroki potrzebne do przekształcenia danych wejściowych w prognozę. Dzięki ML.NET można trenować model niestandardowy, określając algorytm lub importować wstępnie wytrenowane modele TensorFlow i ONNX.

Centralnym elementem w programie korzystającym z biblioteki ML.NET jest obiekt klasy MLContext, jest to singleton koordynujący wszystkie pozostałe elementy, który zapewnia sposoby tworzenia komponentów do przygotowania danych, cech, trenowanie, proces predykcji i ewaluacji modelu.

W przestrzeni ML klasa DataOperationsCatalog jest używana do tworzenia składników, które działają na danych, ale nie są częścią potoku uczenia modelu. Zawiera metody do ładowania, zapisywania, buforowania, filtrowania, mieszania i dzielenia danych. Możliwe jest wczytywanie danych z plików tekstowych

w różnych formatach a także bezpośrednio z większości popularnych systemów bazodanowych. Oprócz tego dostępne są filtry, pozwalają one na wybranie ze zbioru jedynie rekordów spełniających dany warunek. Mamy trzy zasadnicze metody filtracji: FilterRowsByMissingValue - odrzuca wiersze, w których wybrana kolumna nie ma wartości; FilterRowsByColumn - wybiera tylko rekordy, w których wartość wybranej kolumny mieści się w ustalonym przedziale oraz FilterRowsByKeyColumnFraction – wybiera rekordy operując na kluczach. Przygotowanie i wybieranie wierszy danych możliwe jest także dzięki metodom: SkipRaws TakeRows, ShuffleRows. Z kolei CrossValidationSplit dzieli zestaw danych na zestaw treningowy i zestaw testowy, podobnie jak TrainTestSplit.

W klasie TransformsCatalog są metody, które pozwalają na różnego rodzaju transformacje i zmiany wykonywane na danych. Surowe dane muszą być przygotowane lub wstępnie przetworzone, zanim będą mogły zostać użyte do znalezienia parametrów modelu. Mapowanie i grupowanie kolumn jest możliwe dzięki metodom: Concatenate - łączenia jednej lub więcej kolumn wejściowych w nową kolumnę wyjściową, CopyColumns, DropColumns, SelectColumns. Metody do oznaczania i uzupełniania brakujących wartości to: IndicateMissingValues - tworzy nową kolumnę wyjściową logiczną, której wartość jest true, gdy brakuje wartości w kolumnie wejściowej; ReplaceMissingValues - tworzy nową kolumnę wyjściową, której wartość jest ustawiona na wartość domyślną, jeśli wartości nie ma w kolumnie wejściowej. Inne metody umożliwiają przekształcenia danych, na przykład: NormalizeMeanVariance oblicza średnią i wariancję; inne operacje do normalizacji to: NormalizeMeanVariance, NormalizeLogMeanVariance. NormalizeLpNorm - skaluje wektory wejściowe według ich Lp-norm, NormalizeMinMax - skaluje dane w zakresie między minimalną i maksymalną wartością w danych treningowych.

Metody z grupy TransformsCatalog.ConversionTransforms pozwalają konwertować dane np. z wartości tekstowych na reprezentację liczbową, są to: ConvertType - konwertuje typ kolumny wejściowej na nowy typ; Hash mieszanie (haszowanie) wartości w kolumnach wejściowych oraz grupa metod do mapowania na klucze i odwrotnie np. MapValue mapuje wartości do kluczy (kategorii) na podstawie dostarczonego słownika mapowań; MapKeyToVector konwertuje klucze z powrotem na wektory oryginalnych wartości. W grupie TransformsCatalog.CategoricalTransforms mamy OneHotEncoding kodowanie 1 z n jednej lub więcej kolumn tekstu na wektory oraz OneHotHashEncoding kodowanie w oparciu o mieszanie (haszowanie).

W przestrzeni nazw ML.Transforms.TimeSeries umieszczono narzędzia do analizy szeregów czasowych. Dodatkowo dostępne są klasy do transformacji tekstu - TextCatalog oraz przekształcenia obrazu - ImageEstimatorsCatalog oraz klasy z przestrzeni ML.Vision i narzędzia z przestrzeni ML.Transforms.Image.

Cała rodzina klas jest dedykowana podstawowym funkcjonalnościom: AnomalyDetectionCatalog i TimeSeriesCatalog umożliwiają analizę szeregów czasowych, BinaryClassificationCatalog - klasyfikację binarną, ClusteringCatalog -

klasteryzację, MulticlassClassificationCatalog - klasyfikację wieloklasową, RegressionCatalog – regresję. Do każdej z tych klas mamy odpowiednią klasę trenerów. Do trenowania wykrywania anomalii szeregów czasowych mamy metodę RandomizedPca - analizę głównych składowych. Do trenowania klasteryzacji mamy tylko metodę KMeans opartą na algorytmie k-średnich. Do trenowania klasyfikacji binarnej mamy do dyspozycji wiele metod: LbfgsLogisticRegression model liniowej regresji logistycznej trenowany przy pomocy zmodyfikowanego algorytmu Broydena-Fletchera-Goldfarba-Shanno; dwie kolejne metody korzystają z maszyny wektorów podpierających, LdSvm bazuje na modelu Local Deep SVM natomiast LinearSvm, oparta jest na liniowym modelu klasyfikacji binarnej wytrenowanym na danych etykietowanych wartościami logicznymi; SdcaLogisticRegressionBinary oraz SdcaNonCalibratedBinary używają liniowego modelu klasyfikacji opartego o algorytm Stochastic Dual Coordinate Ascent; SgdCalibrated oraz SgdNonCalibrated używają liniowego modelu klasyfikacji opartego o algorytm stochastic gradient descent; FastForestBinary wykorzystuje model klasyfikacji binarnej oparty na drzewie decyzyjnym; GamBinary wykorzystuje uogólnione modele addytywne; FieldAwareFactorizationMachine, wykorzystuje maszynę faktoryzacji wytrenowaną na danych etykietowanych wartościami logicznymi; LightGbmBinary używa wzmocnienia gradientowego dla drzew decyzyjnych; AveragedPerceptron wykorzystuje perceptron dla danych etykietowanych wartościami logicznymi; Prior, bazuje na wcześniejszym modelu klasyfikacji binarnej.

Do trenowania klasyfikacji wieloklasowej również dostępnych jest wiele metod. LightGbm, LbfgsMaximumEntropyMulticlass, NaiveBayes, OneVersusAll, PairwiseCoupling, SdcaMaximumEntropy, SdcaNonCalibrated, oraz ImageClassification, która wykorzystuje głęboką sieć neuronową (DNN) do klasyfikowania obrazów.

Do trenowania regresji można wykorzystać: LightGbm, Ols opartą na metodzie najmniejszych kwadratów (Ordinary least squares), LbfgsPoissonRegression, OnlineGradientDescent, Sdca, FastForest, FastTree oraz Gam.

ML.NET dla każdego modelu oferuje różne zestawy metryk do oceny modelu. Wymienimy głównie nazwy, bez szczegółowych wyjaśnień, są one znaczące a termin angielski nie nastręcza problemu. Dla wykrywania anomalii mamy dwie metryki: AreaUnderRocCurve, DetectionRateAtFalsePositiveCount - stosunek poprawnie zidentyfikowanych anomalii przy określonej liczbie fałszywych trafień. Dla klasteryzacji są trzy metryki: AverageDistance, DaviesBouldinIndex oraz NormalizedMutualInformation. Dla klasyfikacji binarnej są to: Accuracy, AreaUnderPrecisionRecallCurve, AreaUnderRocCurve, ConfusionMatrix, F1Score, NegativePrecision, NegativeRecall, PositivePrecision oraz PositiveRecall. Dla klasyfikacji wielokryterialnej są to ConfusionMatrix, LogLoss, LogLossReduction, MacroAccuracy, MicroAccuracy, PerClassLogLoss,

TopKAccuracy, TopKAccuracyForAllK oraz TopKPredictionCount. Dla modelu regresji są to: LossFunction, MeanAbsoluteError, MeanSquaredError, RootMeanSquaredError. Dodatkowo możliwa jest walidacja krzyżowa, metoda CrossValidate dzieli najpierw zbiór na określoną liczbę podzdbiorów, model jest uczony i ewaluowany dla każdego z nich. Metoda zwraca kolekcję obiektów, z których każdy zawiera model oraz wyznaczone metryki.

WYNIKI EKSPERYMENTU

W celu weryfikacji możliwości ML.NET do budowy modelu jak i wykorzystania go do prognozy, utworzono prostą aplikację w języku C# prognozującą poziom inflacji na podstawie zgromadzonych danych. Celem badania było określenie jak trendy inflacyjne innych państw wpływają na inflację w Polsce przy wykorzystaniu modelu regresji liniowej. Aplikacja trenuje model regresji, na podstawie danych, a następnie ocenia jakość powstałego modelu. Poziom inflacji w Polsce został potraktowany jako zmienna zależna. Utworzony model jest weryfikowany na wybranych danych dla losowo wybranego miesiąca. W badaniu porównano dwa algorytmy regresji: Poissona (LbfgsPoissonRegression) oraz oparty na metodzie optymalizacyjnej Stochastic Dual Coordinate Ascent (SDCA). Ocena jakości modelu polega na wyznaczeniu współczynnika determinacji R^2 dla danych testowych oraz błędu średniokwadratowego. W celu ustalenia zależności zbudowano kolejno modele dla wybranych grup państw.

Algorytm regresji Poissona w klasie LbfgsPoissonRegressionTrainer został zaimplementowany przy pomocy techniki optymalizacji opartej na metodzie ograniczonej pamięci Broydena-Fletchera-Goldfarba-Shanno (L-BFGS) [Liu i in. 1989]. L-BFGS jest metodą quasi-newtonowską, która zastępuje kosztowne obliczenia Hesjanu aproksymacją, ale nadal charakteryzuje się szybkim tempem zbieżności, podobnie jak metoda, w której obliczana jest pełna macierz Hessego [Nash i in. 1991]. Metoda ta nadaje się do zagadnień z wektorami cech o dużych wymiarach. Pierwsze implementacje metody były napisane w Fortranie [Zhu i in. 1997]. Efektywność metody może być poprawiana dzięki regularyzacji [Hardik i in. 2021], implementacja w bibliotece ML.NET pozwala na regularyzację przy pomocy kombinacji norm L1 i L2.

Stochastic Dual Coordinate Ascent to algorytm optymalizacji, w którym maksymalizujemy funkcję dla problemu dualnego. Algorytm sukcesywnie maksymalizuje wzdłuż kierunków współrzędnych. W każdej iteracji algorytm określa współrzędną za pomocą reguły losowego wyboru współrzędnych, a następnie maksymalizuje odpowiednią ustalając wszystkie inne współrzędne [Shalev-Shwartz i in. 2013]. Jest to nowoczesna technika optymalizacji wypukłych funkcji celu, algorytm ten można skalować, ponieważ jest to algorytm uczenia strumieniowego [Tran i in. 2015]. Metoda ma wiele modyfikacji, które poprawiają wydajność [Shalev-Shwartz i in. 2016].

Do analizy wybrano dane z 29 państw łącznie z Polską. Wybrano następujące państwa Wielka Brytania (GB), Irlandia (IE), Niemcy (DE), Francja (FR), Hiszpania (ES), Portugalia (PT), Austria (AT), Włochy (IT), Czechy (CZ), Słowacja (SK), Grecja (GR), Białoruś (BY), Ukraina (UA), Norwegia (NO), Szwecja (SE), Węgry (HU), Finlandia (FI), Rosja (RU), Stany Zjednoczone (US), Chiny (CN), Japonia (JP), Indie (IN), Tajwan (TW), Korea Południowa (KR), ZEA (AE), Katar (QA), Kuwejt (KW), Turcja (TR) oraz Polska (PL). Wszystkie państwa mają kontakty handlowe z Polską i mają wpływ na polską gospodarkę a więc także poziom inflacji. Dla każdego z państw ustalono poziom inflacji dla kolejnych miesięcy 2021 roku (grudzień - prognoza). Dane w trzech grupach przedstawiono w tabeli 1.

Tabela 1. Wartości wskaźnika inflacji w roku 2021 w wybranych państwach

Mies.	GB	IE	DE	FR	ES	PT	AT	IT	CZ	SK
1	0,7%	-0,2%	1,0%	0,6%	0,5%	0,3%	0,8%	0,4%	2,2%	0,7%
2	0,4%	-0,4%	1,3%	0,6%	0,0%	0,5%	1,2%	0,6%	2,1%	0,9%
3	0,7%	0,0%	1,7%	1,1%	1,3%	0,5%	2,0%	0,8%	2,3%	1,4%
4	1,5%	1,1%	2,0%	1,2%	2,2%	0,6%	1,9%	1,1%	3,1%	1,6%
5	2,1%	1,7%	2,5%	1,4%	2,7%	1,2%	2,8%	1,3%	2,9%	2,2%
6	2,5%	1,6%	2,3%	1,5%	2,7%	0,5%	2,8%	1,3%	2,8%	2,9%
7	2,0%	2,2%	3,8%	1,2%	2,9%	1,5%	2,9%	1,9%	3,4%	3,3%
8	3,2%	2,8%	3,9%	1,9%	3,3%	1,5%	3,2%	2,0%	4,1%	3,8%
9	3,1%	3,7%	4,1%	2,2%	4,0%	1,5%	3,3%	2,5%	4,9%	4,6%
10	4,2%	5,1%	4,5%	2,6%	5,4%	1,8%	3,7%	3,0%	5,8%	5,1%
11	5,1%	5,3%	5,2%	2,8%	5,5%	2,6%	4,3%	3,7%	6,0%	5,6%
12	5,4%	5,3%	5,4%	3,0%	5,7%	2,6%	4,3%	3,9%	6,7%	5,6%

Mies.	GR	BY	UA	NO	SE	HU	FI	RU	US	CN
1	-2,0%	7,7%	5,0%	2,5%	1,6%	2,7%	0,9%	5,2%	1,4%	-0,3%
2	-1,3%	8,7%	6,1%	3,3%	1,4%	3,1%	0,9%	5,7%	1,7%	-0,2%
3	-1,6%	8,5%	7,5%	3,1%	1,7%	3,7%	1,3%	5,8%	2,6%	0,4%
4	-0,3%	8,7%	8,5%	3,0%	2,2%	5,1%	2,1%	5,5%	4,2%	0,9%
5	0,1%	9,4%	8,4%	2,7%	1,8%	5,1%	2,2%	6,0%	5,0%	1,3%
6	1,0%	9,9%	9,5%	2,9%	1,3%	5,3%	2,0%	6,5%	5,4%	1,1%
7	1,4%	9,8%	9,5%	3,0%	1,4%	4,6%	1,9%	6,5%	5,4%	1,0%
8	1,9%	9,8%	10,2%	3,4%	2,1%	4,9%	2,2%	6,7%	5,3%	0,8%
9	2,2%	10,2%	10,2%	4,1%	2,5%	5,5%	2,5%	7,4%	5,4%	0,7%
10	3,4%	10,5%	11,0%	3,5%	2,8%	6,5%	3,2%	8,1%	6,2%	1,5%
11	4,8%	10,3%	10,9%	5,1%	3,3%	7,4%	3,7%	8,4%	6,8%	2,3%
12	5,0%	10,8%	10,3%	4,8%	3,3%	7,2%	3,7%	8,4%	6,9%	2,6%

Mies.	JP	IN	TW	KR.	AE	QA	KW	TR	PL
1	-1,2%	4,1%	-0,2%	0,6%	-2,4%	-1,3%	2,2%	15,0%	2,6%
2	-0,7%	5,0%	1,4%	1,1%	-2,1%	-1,4%	2,0%	15,6%	2,4%
3	-0,5%	5,5%	1,2%	1,5%	-1,9%	-0,3%	2,5%	16,2%	3,2%
4	-0,4%	4,2%	2,1%	2,3%	-2,0%	1,0%	2,8%	17,1%	4,3%
5	-1,1%	6,3%	2,4%	2,6%	-1,1%	2,5%	3,0%	16,6%	4,7%
6	-0,8%	6,3%	1,8%	2,4%	-0,5%	2,0%	3,0%	17,5%	4,4%
7	-0,5%	5,6%	1,9%	2,6%	-0,4%	3,1%	3,0%	19,0%	5,0%
8	-0,3%	5,3%	2,3%	2,6%	-0,5%	3,0%	3,2%	19,3%	5,5%
9	-0,4%	4,4%	2,6%	2,5%	0,0%	2,7%	3,1%	19,6%	5,9%
10	0,2%	4,5%	2,5%	3,2%	0,6%	4,3%	3,2%	19,9%	6,8%
11	0,1%	4,9%	2,8%	3,7%	1,2%	6,1%	3,5%	21,3%	7,4%
12	0,4%	4,9%	2,9%	3,7%	1,3%	6,1%	3,5%	23%%	8,1%

Źródło: opracowanie własne na podstawie https://tradingeconomics.com

W tabeli 2 przedstawiono wyniki eksperymentu.

Tabela 2. Wynik dla wszystkich wybranych państw

	Metoda	Współczynnik determinacji	Pierwiastek błędu średniokwadratowego	Estymowana inflacja w Polsce	Rzeczywista inflacja w Polsce
28 państw	Regresja Poissona	0,98	0,22	7,6073%	7,4%
	SDCA	1,0	0,01	7,4135%	7,4%
Strefa Euro	Regresja Poissona	0,97	0,3	7,7047%	7,4%
	SDCA	1,0	0,11	7,5996%	7,4%
Strefa Euro + GB	Regresja Poissona	0,97	0,28	7,6734%	7,4%
	SDCA	1,0	0,1	7,6003%	7,4%
UE	Regresja Poissona	0,97	0,28	7,6806%	7,4%
	SDCA	1,0	0,09	7,5258%	7,4%
Silne gospodarki	Regresja Poissona	0,97	0,28	7,4634%	7,4%
	SDCA	1,0	0,08	7,5928%	7,4%
Silne militarnie	Regresja Poissona	0,97	0,32	7,522%	7,4%
	SDCA	0.98	0,22	7,621%	7,4%
Sąsiedzi z UE	Regresja Poissona	0,95	0,39	7,6054%	7,4%
	SDCA	0.82	0,75	8,1375%	7,4%
Zatoka Perska	Regresja Poissona	0,94	0,42	8,0765%	7,4%
	SDCA	0.96	0,33	7,6698%	7,4%

Źródło: opracowanie własne

Kolejno uwzględniono wpływ wszystkich 28 państw na inflację w Polsce. Następnie wybrano podgrupę państw ze strefy Euro (Irlandia, Niemcy, Francja,

Hiszpania, Portugalia, Austria, Włochy, Słowacja, Grecja, Finlandia) oraz strefę Euro i Zjednoczone Królestwo. Kolejną czwartą, badaną podgrupą były kraje Unii Europejskiej (Irlandia, Niemcy, Francja, Hiszpania, Portugalia, Austria, Włochy, Czechy, Słowacja, Grecja, Szwecja, Węgry, Finlandia). Piątą grupą były państwa o dużym potencjale gospodarczym (Niemcy, Francja, Rosja, USA, Chiny, Japonia, Indie, Tajwan, Korea Południowa). Szóstą podgrupę stanowiły państwa o dużym potencjale militarnym (Rosja, USA, Chiny, Indie). Siódmą grupę stanowili sąsiedzi z Unii Europejskiej (Niemcy, Czechy, Słowacja). Na zakończenie sprawdzono grupę państw z rejonu Zatoki Perskiej (ZEA, Katar, Kuwejt).

PODSUMOWANIE

ML.NET umożliwia integrację algorytmów uczenia maszynowego z praktycznymi aplikacjami. Przygotowanie aplikacji w języku C# nie jest trudne, w prosty sposób można wygenerować model i zastosować go do predykcji. Wspomagające narzędzie Model Builder pozwalające na automatyczną generację modelu okazało się z kolei niezbyt przydatne. Model Builder tylko w szczególnych przypadkach generuje model, proces często kończy się błędem. W przypadku rozważanej regresji, podczas generacji sprawdzana jest kolejno każda dostępna metoda, i finalnie wybierana ta która ma najwyższy współczynnik determinacji. Niestety błąd którejkolwiek metody, spowodowany specyfiką danych (np. zbyt mało danych) powoduje przerwanie autogeneracji, nie ma możliwości predefiniowania tylko wybranych metod. Innym problemem jest kwestia wydajności. W przypadku niewielkiego zbioru danych wybranego w eksperymencie czas trenowania modelu jest krótki, jednakże w pracy [Kędziora i in. 2021] wyniki dla biblioteki ML.NET były znacznie gorsze niż dla TensorFlow.

BIBLIOGRAFIA

Ahmed Z. et al. (2019) Machine Learning at Microsoft with ML.NET. Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, n. pag.

Capellman J. (2020) Hands-On Machine Learning with ML.NET: Getting Started with Microsoft ML.NET to Implement Popular Machine Learning Algorithms in C#. Packt Publishing, Birmingham.

Gevorkyan M. N., Demidova A. V., Demidova T. S., Sobolev A. A. (2019) Review and Comparative Analysis of Machine Learning Libraries for Machine Learning. Discrete and Continuous Models and Applied Computational Science, 27(4), 305-315, doi: 10.22363/2658-4670-2019-27-4-305-315.966-979.

Kędziora E. J. Maksim G. K. (2021) Analiza wydajności bibliotek uczenia maszynowego. Journal of Computer Sciences Institute, 20, 230-236.

Liu D. C., Nocedal J. (1989) On the Limited Memory BFGS Method for Large Scale Optimization. Mathematical Programming, 45, 503-528.

Nash S. G., Nocedal J. (1991) A Numerical Study of the Limited Memory BFGS Method and the Truncated-Newton Method for Large Scale Optimization. SIAM Journal of Optimization, 1, 358-372.

Shalev-Shwartz S., Zhang T. (2013) Stochastic Dual Coordinate Ascent Methods for Regularized Loss Minimization. Journal of Machine Learning Research, 14, 567-599.

Shalev-Shwartz S., Zhang T. (2016) Accelerated Proximal Stochastic Dual Coordinate Ascent for Regularized Loss Minimization. Mathematical Programming, 155, 105-145.

Tankaria H., Sugimoto S., Yamashita N. (2021) A Regularized Limited Memory BFGS Method for Large-Scale Unconstrained Optimization and its Efficient Implementations. arXiv: Optimization and Control, n. pag.

Tran K., Hosseini S., Xiao L., Finley T., Bilenko M. (2015) Scaling up Stochastic Dual Coordinate Ascent. Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, n. pag.

Zhu C. Byrd R. H., Lu P., Nocedal J. (1997) Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. ACM Transactions on Mathematical Software, 23(4), 550-560, doi:10.1145/279232.279236.

https://dotnet.microsoft.com/en-us/apps/machinelearning-ai/ml-dotnet [dostęp: 20.11.2021].

APPLICATION OF THE ML.NET LIBRARY TO ECONOMIC ISSUES

Abstract: The aim of the paper was to attempt to validate the suitability of the ML.NET library for research in the field of economics. The possibilities of using machine learning in applications are presented, and the free programming libraries related to machine learning are briefly discussed. The functionality of the ML.NET library, with particular emphasis on its suitability for research and applications in the field of economics, was discussed. In order to verify the possibility of ML.NET to the model generation and its use for forecasting, a simple application in C# language to forecast the level of inflation based on the collected data was created.

Keywords: machine learning, ML.NET, application of computer science in economics

JEL classification: C80