Wydział Matematyki
i Nauk Informacyjnych
POLITECHNIKA WARSZAWSKA

Automatyczne uczenie maszynowe

Tworzenie modelu klasyfikacji

Jakub Brzóskowski 313180, Jakub Knyspel 313282

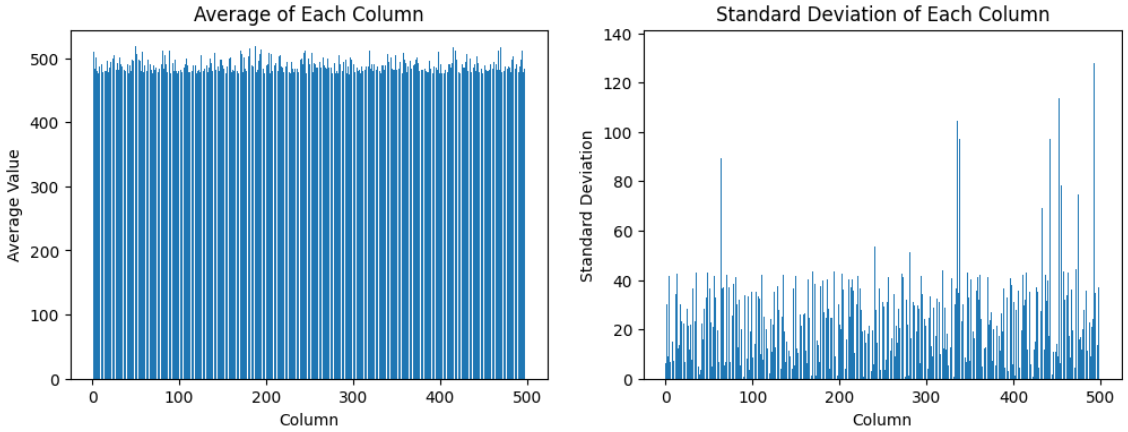
16 stycznia 2024

1 Wstęp

Celem zadania jest utworzenie dwróch modeli klasyfikacji dla zadanego zbioru danych. Pierwszy z modeli jest tworzony manualnie - z ręcznym wyborem typu modelu oraz hiperparametrów. Drugi z modeli jest tworzony za pomocą automatycznego uczenia maszynowego. Całość została opracowana przy użyciu modułów dostępnych w języku Python.

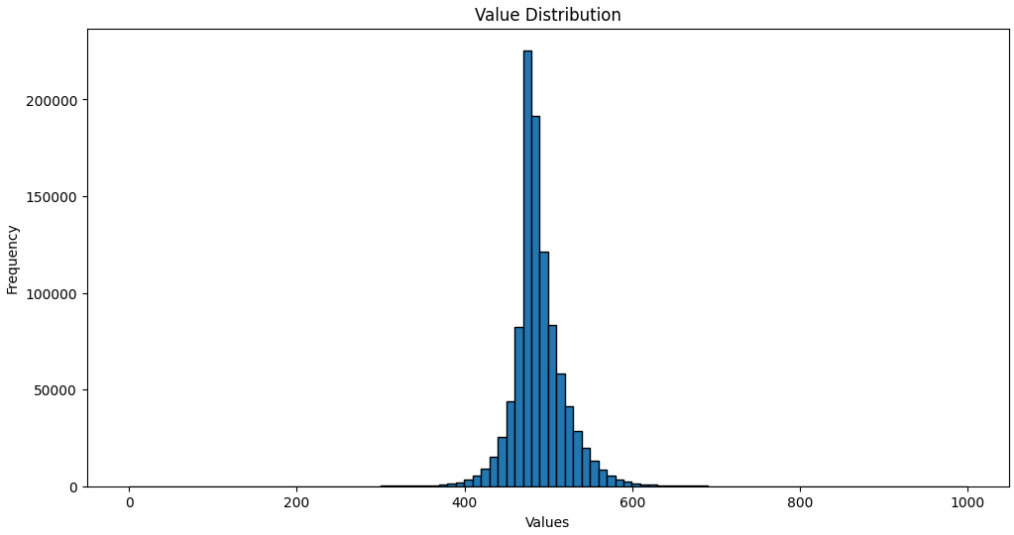
2 Analiza zbioru danych

Przed przstąpieniem do tworzenia modeli przeprowadziliśmy wstępną analizę danych. Zadany zbiór to 2000 próbek składających się z 500 cech (kolumn). Wartości wszystkich kolumn to liczby całkowite z zakresu 0 - 999. Średnie wartości kolumn nie odbiogają od siebie znacząco, natomist odchylenia standardowe znacząco się różnią. Wartości te przedstawione są na wykresie 1.



Rysunek 1: Średnie wartości i odchylenia standardowe cech

Dystrybucja wartości jest silnie skoncentrowana w okolicy wartości średniej (w przedziale 400 - 600 mieści się 98.8% wartości). Histogram wartości przedstawiony jest na wykresie 2.

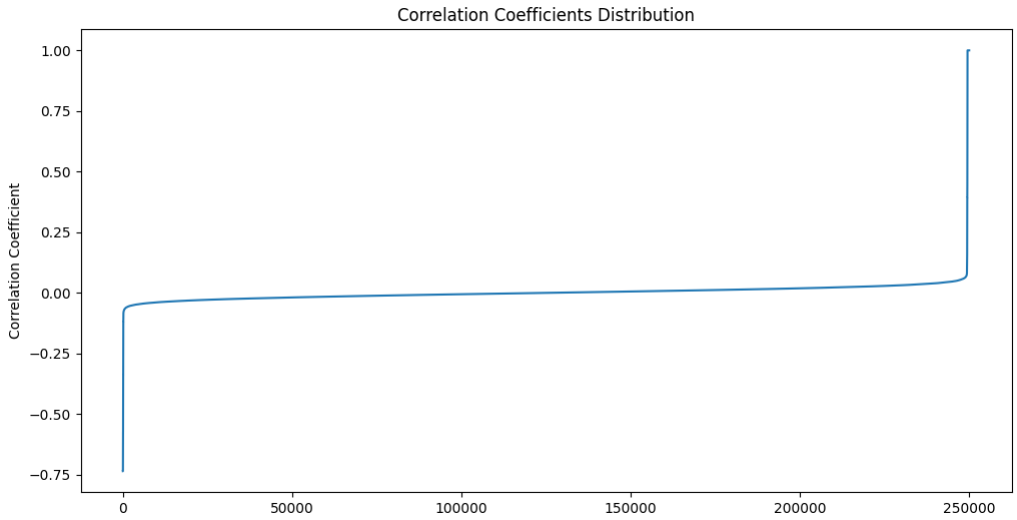


Rysunek 2: Histogram wartości

Próbki są przypisane do dwóch klas - "1"oraz 1". Zbiór zawiera 1000 próbek klasy "1", i 1000 próbek klasy 1".

2.1 Korelacja cech

W celu dalszej analizy danych, obliczyliśmy współczynnik korelacji pomiędzy parami kolumn za pomocą metody Pearsona. Wykres 3 przedstawia dystrybucję korelacji.



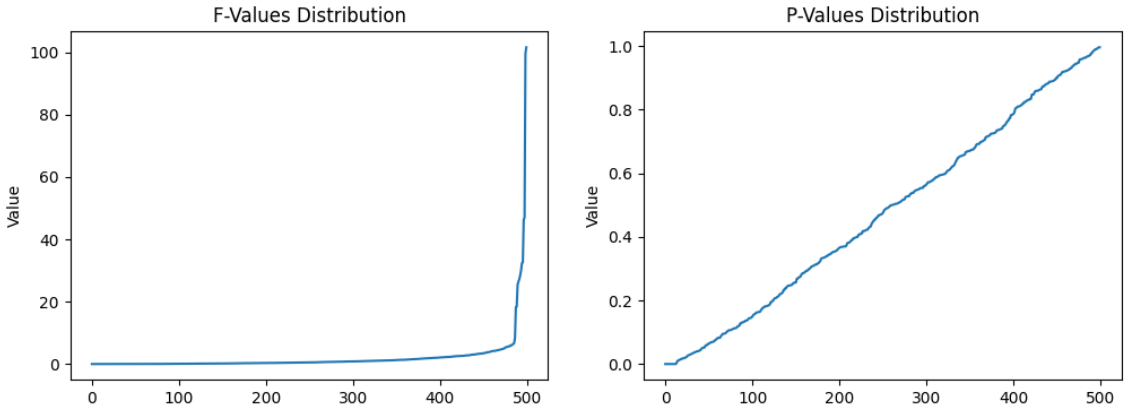
Rysunek 3: Rozkład współczynników korelacji

Jak widać, zdecydowana większość par nie wykazuje żadnej korelacji. Jednak w niektórych przypadkach obserwujemy silną korelację lub antykorelację. Sugeruje to, że niektóre cechy są ze sobą blisko związane i nie niosą żadnej dodatkowej informacji. Powinniśmy zatem je odrzucić przed procesem uczenia.

Przed uczeniem modeli zdecydowaliśmy się na odrzucenie wszystkich kolumn, których współczynnik korelacji z inną kolumną wynosi (co do wartości bezwzględnej) > 0.8. Oczywiście z każdej takiej grupy skorelowanych kolumn pozostawiliśmy jedną.

2.2 Wybór najlepszych cech

Zbiór danych poddaliśmy analizie wariancji. Dla każdej cechy w zbiorze danych przeprowadzony został F-test ANOVA. Dokładniej: test porównywał średnie wartości cechy w próbkach należących do klasy 1, do średnich wartości cechy w klasie -1. Wyniki tych testów przedstawione są na wykresach 4. Wykres z lewej strony przedstawia F-wartość, a wykres z prawej - p-wartość.



Rysunek 4: Rozkład F-wartości i P-wartości otrzymanych z testu ANOVA

Z wykresów wynika, że cechy są w różnym stopniu powiązane z przynależnością do przewidywanych grup - część nie ma z tym żadnego związku, a część jest z przynależnością bardzo silnie powiązana. Sugeruje to, że proces uczenia powinien być przeprowadzony tylko na części cech.

Przed uczeniem modeli zdecydowaliśmy się na odrzucenie wszystkich kolumn poza 100 najsilniej powiązanymi z przynależnością do danej klasy.

3 Model tworzony manualnie

Ze wstępnej analizy zbioru danych wynika, że zawiera on dużo cech (w porównaniu do ilości próbek), z których niektóre są bardzo powiązane z klasą przynależności, a niektóre nie są z nią powiązane. Oznacza to, że najbardziej obiecującymi modelami do tego zadania są modele bazujące na drzewach decyzyjnych.

Jednakże, jeśli naszym celem jest znalezienie jak najlepszego modelu, zdecydowaliśmy się przetestować także inne rodzaje modeli. Dla każdego modelu przeprowadziliśmy manualny tuning hiperparametrów. Wypróbowane modele to:

- Las losowy (`RandomForestClassifier` z modułu `scikit-learn`)
- k Nearest Neighbors (`KNeighborsClassifier`)
- Drzewo decyzyjne (`DecisionTreeClassifier`)
- Sieć neuronowa (`MLPClassifier`)
- Gradient Boosting (`GradientBoostingClassifier`)
- Regresja logistyczna (`LogisticRegression`)
- Drzewo decyzyjne z meta-estymatorem *AdaBoost* (`DecisionTreeClassifier` i `AdaBoostClassifier`)
- *Support Vector Classifier* (`SVC`)
- *Gaussian Process Classifier* (`GaussianProcessClassifier`)

3.1 Dalsza optymalizacja

Wyniki przedstawione powyżej nie są zadowalające. Aby poprawić dokładność przewidywania, zastosowaliśmy meta-estymator *Ensemble*, który przewiduje klasyfikację poprzez głosowanie pewnego

zbioru estymatorów. Do utworzenia *Ensemble* wykorzystaliśmy wszystkie podane powyżej estymatory z wyjątkiem *AdaBoost*.

Dodatkowo przeprowadziliśmy tuning hiperparametrów modeli będących częścią *Ensemble*. Wykorzystaliśmy do tego metodę losowego przeszukiwania przestrzeni parametrów.

Ostateczny model osiąga dokładność **0.846** na zbiorze testowym. Estymatory tworzące *Ensemble* wraz z najlepszymi parametrami to:

- Las losowy (`RandomForestClassifier`),
  Liczba estymatorów = 200, Maksymalna wysokość drzewa = 30
- k Nearest Neighbors (`KNeighborsClassifier`), Liczba sąsiadów = 3
- Drzewo decyzyjne (`DecisionTreeClassifier`), Maksymalna wysokość drzewa = 10
- Sieć neuronowa (`MLPClassifier`), 1 x 100 neuronów, $\alpha = 0.001$
- Gradient Boosting (`GradientBoostingClassifier`),
  Liczba estymatorów = 50, Learning rate = 0.01
- Regresja logistyczna (`LogisticRegression`), $C = 0.01$
- *Support Vector Classifier* (`SVC`), Kernel - *Radial basis function*, $C = 100$
- *Gaussian Naive Bayes* (`GaussianNB`)

4 Model tworzony automatycznie

Model automatycznego uczenia maszynowego został utworzony przy użyciu modułu $auto-sklearn$. Nie wymaga on podawania praktycznie żadnych parametrów z wyjątkiem opcji dotyczących czasu pracy i zasobów (ilości pamięci RAM, ilości wątków, itp.), które mogą zostać wykorzystane w procesie optymalizacji.

Jedynymi parametrami związnaymi z samym procesem optymalizacji to metoda oceniania jakości modelu, oraz strategia resamplingu (zapobiegania *overfittingowi*). W pierwszym przypadku jest to *balanced accuracy*, a w drugim - walidacja skrośna przy podziale zbioru treningowego na 10 części.

Po 30-minutowym procesie optymalizacji, model uzyskuje dokładność **0.878** na zbiorze testowym.

5 Podsumowanie

Zarówno model tworzony ręcznie, jak i model tworzony automatycznie, dają satysfakcjonujące wyniki. Co ciekawe, model automatyczny daje niewiele lepsze wyniki, mimo korzystania z podobnych technik (te same bazowe estymatory oraz *Ensembling*). Z drugiej strony interesujące jest, jak minimalna konfiguracja modelu automatycznego wystarczyła do uzyskania takiego wyniku.

6 Bibliografia

- Dokumentacja *scikit-learn* - https://scikit-learn.org/stable/
- Dokumentacja *auto-sklearn* - https://automl.github.io/auto-sklearn/master/index.html