Modele liniowe

Celem zadania jest poeksperymentowanie z zadaniami regresji dla stosunkowo prostych zbiorów danych (nie chcemy przeprowadzać skomplikowanej inżynierii cech, a raczej zobaczyć, jak działają modele i jak możemy poprawiać ich jakość).
Potencjalnie pomocne biblioteki: pandas, scikit-learn, numpy, matplotlib, seaborn
Nie przerażcie się rozmiarem instrukcji - starałam się dokładnie opisać poszczególne punkty, ale pracy nie jest aż tak dużo ;)
Wynikiem zadania powinien być raport w formacie .pdf i paczka .zip z kodem, z pomocą którego wykonaliście zadania. Raport niech będzie *rozsądnie* długi: powinien zawierać udokumentowane kolejne kroki, w szczególności obserwacje, wnioski, odpowiedzi na pytania. Nie wrzucajcie wszystkiego, unikajcie wrzucania kodu (choć czasem może się przydać, ale z rozsądkiem). Wykresy i tabelki mile widziane.
W przypadku pytań proszę o kontakt. Miłej zabawy!

1. Regresja liniowa
    a. Załaduj zbiór California housing (jest on dostępny w scikit-learn). Zapoznaj się z danymi i znaczeniem kolumn.
    b. Narysuj wykres *pairplot* zależności cech (na osiach dwie cechy, kolor w zależności od ceny mieszkania - MedHouseVal). Konieczne będzie stworzenie osobnych “kategorii” w zależności od ceny - może się przydać funkcja pandas.qcut, polecam skorzystanie z kodu:
    `df["MedHouseVal"] = pd.qcut(df["MedHouseVal"], 6, retbins=False)`
    `df["MedHouseVal"] = df["MedHouseVal"].apply(lambda x: x.mid)`
    `sns.pairplot(data=df, hue="MedHouseVal")`
    Zbadaj korelacje pomiędzy cechami. Obejrzyj histogramy dla każdej cechy. Skomentuj wyniki. Co powiesz o zmiennych AveBedrms, AveRooms, Population i AveOccup?
    c. Usuń outliery z danych. Skorzystaj z poniższej linijki:
    `df = df.loc[(df["AveBedrms"] < 4) & (df["AveRooms"] < 21) & (df["AveOccup"] < 9) & (df["Population"] < 8000)]`
    Obejrzyj jeszcze raz dane. Czy masz przeczucie, które zmienne mogą okazać się najważniejsze podczas regresji?
    d. Wyodrębnij kolumnę MedHouseVal, przypisz ją do nowej zmiennej i usuń z reszty zbioru.
    e. Podziel dane na treningowe i testowe w proporcji 80%-20% z wykorzystaniem train_test_split. Ustaw random_state=0.
    f. Wykorzystaj MinMaxScaler do normalizacji wartości zmiennych. Pamiętaj, że poprawne wykonanie tego punktu zakłada dopasowanie skalera wyłącznie do danych treningowych! Transformujemy oba zbiory (train i test). Sprawdź, jak teraz wyglądają dane (wystarczy histogram).
    g. Dopasuj model regresji liniowej (sklearn.linear_model.LinearRegression) do zbioru treningowego i dokonaj ewaluacji na zbiorze testowym. Zbadaj błąd MAE (nie chcemy bardzo karać za outliery, dlatego nie RMSE). Porównaj go do średniej wartości zmiennej zależnej (czyli MedHouseVal). Czy ten błąd jest duży?

h. Przepuść dane treningowe przez model i sprawdź błąd. Porównaj błąd na danych treningowych i testowych. Czy mamy do czynienia z under- albo overfittingiem?
i. Sprawdź, którym cechom model przypisał największe wagi. Czy wynik pokrywa się z Twoimi przypuszczeniami z punktu c)?
j. Spróbujmy dołożyć nowe cechy do modelu (regresja wielomianowa). Wykorzystaj sklearn.preprocessing.PolynomialFeatures. Użyj kodu:
poly_features = PolynomialFeatures(degree=3, include_bias=False)
(nie dodajemy biasu, bo regresja liniowa sama go doda). Puść fit tylko na danych treningowych, a transform na obu zbiorach. Sprawdź, ile teraz jest zmiennych.
k. Dopasuj model regresji liniowej do nowych danych, zbadaj MAE na danych treningowych i testowych. Skomentuj wynik. Co się stanie, jeśli zwiększymy degree do 5?
l. Wykorzystanie degree=5 jest jak widać zdecydowanie przesadzone, ale spróbujmy na tym przykładzie dokonać regularyzacji. Na początku zbadaj wagi modelu: narysuj histogram, sprawdź wartość maksymalną i minimalną, a także, czy jakieś wagi są równe zero.
m. Wytrenuj modele Ridge oraz Lasso z sklearn.linear_model. Inicjalizując je, ustaw random_state=0. Zwróć uwagę na parametr alpha modeli. Wytrenuj oba modele na domyślnych wartościach parametrów. Zbadaj MAE modeli na obu zbiorach. Dla modelu Ridge sprawdź również histogram wag oraz czy jakieś wagi się wyzerowały. Skomentuj wyniki.
n. Dla modelu Lasso uzyskaliśmy underfitting (co o tym świadczy?), spróbujmy więc dopasować lepszy model. Wykorzystaj model LassoCV i ustaw alpha=100 (w ten sposób sprawdzimy 100 różnych wartości alpha). Ustaw cv równe 5 - doczytaj, za co odpowiada ten parametr. Ponownie random_state=0.
o. Wytrenuj model na danych treningowych. Jaka wartość alpha pozwoliła uzyskać najlepsze wyniki? Zbadaj MAE i wagi jak wcześniej. Jaki procent cech zostanie wyeliminowanych?
p. Który model najlepiej sobie poradził? Podsumuj zadanie.

2. Regresja logistyczna
   a. Załaduj zbiór danych palmerpenguins (plik penguins.csv, więcej informacji nt. tego zbioru znajdziesz tutaj: https://allisonhorst.github.io/palmerpenguins/). Zapoznaj się z danymi i znaczeniem kolumn, zwróć uwagę na brakujące wartości. Przyda się tutaj biblioteka pandas.
   b. W ramach tego zadania nie będziemy się przejmować wierszami z brakującymi wartościami. Usuń wiersze zawierające brakujące dane.
   c. Usuń kolumnę z rokiem.
   d. Wartością, którą będziemy chcieli przewidywać, jest kolumna “species”. Sprawdź, ile wierszy jest przypisanych do każdej kategorii. Czy widzisz jakiś problem?
   e. Narysuj wykresy *pairplot* (może się przydać biblioteka seaborn) zależności cech (na osiach dwie cechy, kolor w zależności od gatunku). Skomentuj wyniki. Może Ci tutaj również pomóc analiza korelacji zmiennych.

f. Celem zadania będzie wykonanie klasyfikacji binarnej. Do tego zadania wybierzemy sobie pingwiny z dwóch klas: Adele i Chinstrap, które wydają się być trudniejsze do oddzielenia od siebie, niż dowolna z tych dwóch i Gentoo. Wyodrębnij te dane ze zbioru.
g. Zbadaj, na jakich wyspach zostały zaobserwowane pingwiny z interesujących nas klas. Z której wyspy pochodzą wszystkie pingwiny z gatunku Chinstrap? Nie chcemy ułatwiać naszemu modelowi zadania, więc usuńmy tę kolumnę.
h. Wróć do przygotowanych wcześniej wykresów *pairplot*, patrząc jedynie na interesujące nas dwie klasy. Czy masz przeczucie, które zmienne mogą okazać się najważniejsze podczas klasyfikacji?
i. Przekonwertuj płeć na wartości 0, 1. Odpowiedniejszą nazwą dla nowej zmiennej byłoby teraz is_male/is_female, ale nie będziemy się aż tak przejmować nazewnictwem.
j. Będziemy mieli do czynienia z problemem klasyfikacji niezbalansowanej. Na szczęście funkcja kosztu w regresji logistycznej pozwala na dodanie wag klas (class weights), aby przypisać większą wagę słabiej reprezentowanej klasie. Scikit-learn dla wartości class_weights="balanced" obliczy wagi odwrotnie proporcjonalne do częstości danej klasy w zbiorze.
k. Wyodrębnij kolumnę ‘species’, która będzie wartością, którą chcemy przewidywać. Zapisz ją w osobnej zmiennej i usuń z reszty danych.
l. Podziel dane na treningowe i testowe w proporcji 80%-20%. Zachowaj proporcje klas (w scikit-learnie będzie to parametr stratify przy wykorzystaniu train_test_split). Ustaw random_state=0.
m. Wykorzystaj StandardScaler do standaryzacji zmiennych numerycznych (wszystkie poza “sex”). Pamiętaj, że poprawne wykonanie tego punktu zakłada dopasowanie skalera wyłącznie do danych treningowych! Transformujemy oba zbiory (train i test).
n. Dopasuj model regresji logistycznej (sklearn.linear_model.LogisticRegression) do danych. Ustaw brak regularyzacji. Pamiętaj o ustawieniu zbalansowania klas!
o. Przetestuj model na danych testowych i sprawdź, na ilu przykładach model się pomylił. Skomentuj wynik.
p. Zbadaj, które zmienne miały największy wpływ na wynik, poprzez sprawdzenie wag modelu. Czy zgadza się to z Twoimi wcześniejszymi przypuszczeniami?
q. Usuń zmienną, do której przypisana została największa waga i wytrenuj model jeszcze raz. Skomentuj wyniki.
r. Wybierz teraz dwie cechy, co do których spodziewasz się, że wystarczą, by uzyskać dobre wyniki (niech to nie będą obie cechy, którym pierwszy model przypisał największe wagi). Wytrenuj na nich model. Skomentuj wybór zmiennych i wyniki.
s. Wejdźmy teraz nieco głębiej do wytrenowanego w poprzednim punkcie modelu. Pobierz wagi z modelu i przemnóż je odpowiednio przez zbiór testowy zawierający wybrane w poprzednim kroku cechy. Dodaj do wyniku wyraz wolny. Dokończ klasyfikację poprzez nadanie odpowiednich etykiet przykładom, dla których wynik >= 0 oraz pozostałym. Czy wyniki pokrywają się z uzyskanymi poprzez wywołanie funkcji predict()?

t. Zaprezentuj wyniki klasyfikacji na wykresie. Niech na dwóch osiach znajdą się wybrane cechy, a sam wykres pokaże punkty odpowiednio pokolorowane na dwa kolory (sprawdzamy, jak model przyporządkował przykłady, nie uwzględniamy prawdziwych klas). Skomentuj wynik. Co powiesz o granicy decyzyjnej?
u. Przekonwertuj wynik podpunktu s) z pomocą funkcji sigmoid. Porównaj wynik z predict_proba modelu.