Projekt 1 - Rozpoznanie ile zarabia osoba

Kacper Hołowaty

15 maja 2024

1 Wstęp

W projekcie zdecydowałem, że będę korzystał z bazy dostępnej tutaj: Adult Census Income przedstawiającą dane odnośnie zarobków osób (czy są powyżej, czy poniżej 50k), w zależności od ich pochodzenia, stanu cywilnego, płci itp. Celem projektu było sprawdzenie różnych technik/modeli klasyfikacji, próbując osiągnąć jak najwyższy wynik klasyfikacji.

2 Baza Danych

Krótki opis wszystkich kolumn bazy (tylko te, z których skorzystałem)

- **age**: Wiek osoby
- **workclass**: Rodzaj zatrudnienia
- **education.num**: Poziom edukacji przedstawiony liczbowo
- **marital.status**: Stan cywilny
- **occupation**: Zawód wykonywany przez osobę
- **relationship**: Związek z inną osobą w rodzinie
- **race**: Rasa osoby
- **sex**: Płeć osoby
- **hours.per.week**: Liczba godzin pracy w tygodniu
- **native.country**: Kraj pochodzenia
- **income**: Dochód roczny

Zgodnie z tym co napisałem poprzednio, celem jest rozpoznanie ile zarabia osoba (czyli przewidywanie wartości kolumny income).

3 Preprocessing

Zauważyłem, że w bazie, występuje sporo brakujących danych. Są one oznaczone "?". Poniższy kod przekształca te dane najpierw na wartości None, a następnie uzupełnia te puste miejsca wartościami najczęściej występującymi w danej kolumnie. Możemy tak zrobić, gdyż "?"występują jedynie w kolumnach z danymi typu string.

```
df.replace("?", pd.NA, inplace=True)
print(df.isnull().sum())
for column in df.columns:
    if df[column].dtype == 'object':
        most_common_value = df[column].mode()[0]
        df[column].fillna(most_common_value, inplace=True)
```

Kolejnym krokiem było zamienienie wszystkich wartości nie liczbowych na liczbowe. Oraz następnie wyodrębnienie tych kolumn, których będę używać do klasyfikacji.

```
df['sex'] = df['sex'].map({'Male': 1, 'Female': 0})
df['marital.status'] = df['marital.status'].replace(['Never-married', 'Divorced', 'Separated', 'Widowed'], 'Single')
df['marital.status'] = df['marital.status'].replace(['Married-civ-spouse', 'Married-spouse-absent', 'Married-AF-spouse'], 'Married')
df['marital.status'] = df['marital.status'].map({'Married': 1, 'Single': 0})

df['occupation'] = df['occupation'].replace(['Prof-specialty', 'Exec-managerial', 'Tech-support'], 'White-collar')
df['occupation'] = df['occupation'].replace(['Craft-repair', 'Machine-op-inspct', 'Transport-moving', 'Handlers-cleaners', 'Farming-fishing'], 'Blue-collar')
df['occupation'] = df['occupation'].replace(['Adm-clerical', 'Sales', 'Other-service', 'Protective-serv', 'Priv-house-serv', 'Armed-Forces'], 'Service')
df['occupation'] = df['occupation'].replace(['White-collar','Blue-collar','Service'],[0,1,2])

df['relationship'] = df['relationship'].replace(['Husband', 'Wife', 'Own-child'], 'Family')
df['relationship'] = df['relationship'].replace(['Not-in-family', 'Unmarried', 'Other-relative'], 'Non-Family')
df['relationship'] = df['relationship'].map({'Family': 1, 'Non-Family': 0})

df['race'] = df['race'].replace(['White'], 'White')
df['race'] = df['race'].replace(['Black', 'Asian-Pac-Islander', 'Amer-Indian-Eskimo', 'Other'], 'Non-White')
df['race'] = df['race'].map({'White': 1, 'Non-White': 0})

df.loc[df['native.country'] != 'United-States', 'native.country'] = 'Other'
df['native.country'] = df['native.country'].map({'United-States': 1, 'Other': 0})

df['income'] = df['income'].map({'<=50K': 0, '>50K': 1})

columns = ['age', 'education.num', 'marital.status', 'occupation', 'relationship', 'race', 'sex', 'hours.per.week', 'native.country', 'income']
df = df[columns]
```

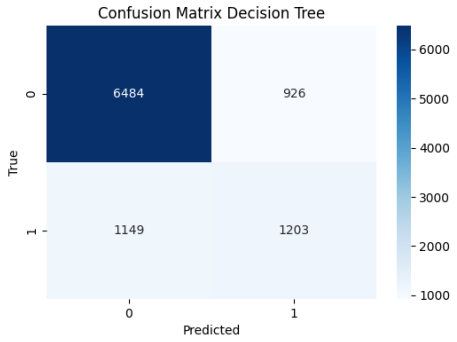
4 Klasyfikacja

Zanim zaczniemy klasyfikację, dzielę dane na zbiór treningowy i testowy.

```
X = df.drop(['income'], axis=1)
y = df['income']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
```

Drzewo decyzyjne

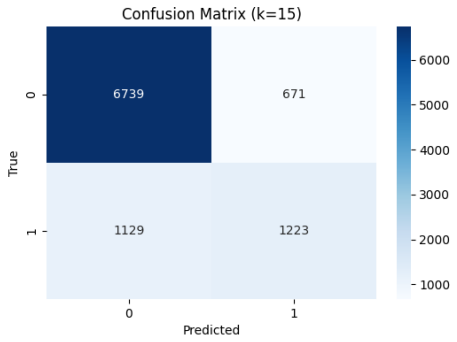
Korzystając z klasyfikatora drzewa decyzyjnego, otrzymuję dokładność (w przybliżeniu): 78.74%



Rysunek 1: Macierz błędu dla DTC

Klasyfikacja metodą k-najbliższych sąsiadów

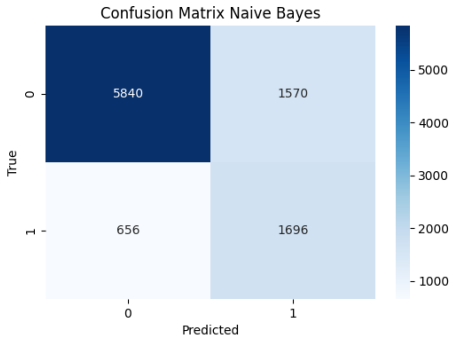
Korzystając z klasyfikacji metodą k-najbliższych sąsiadów, dla k=15 otrzymuję dokładność (w przybliżeniu): 81.56%



Rysunek 2: Macierz błędu dla KNN, gdzie k=15

Klasyfikator Naive Bayes

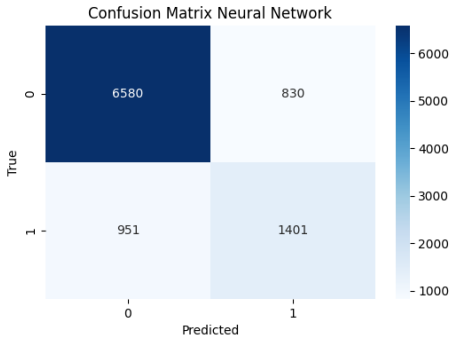
Korzystając z klasyfikatora Naive Bayes, otrzymuję dokładność (w przybliżeniu): 77.20%



Rysunek 3: Macierz błędu dla klasyfikatora Naive Bayes

Klasyfikator MLP (Multilayer Perceptron)

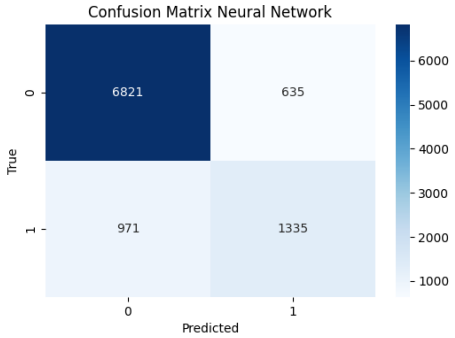
Korzystając z jednokierunkowej sieci neuronowej, czyli wielowarstwowego perceptronu (Multilayer Perceptron, MLP), dla sieci z dwiema warstwami neuronowym, jedna ma 3 neurony, druga ma 2 neurony, otrzymuję dokładność (w przybliżeniu): 81.76%



Rysunek 4: Macierz błędu dla MLP

Klasyfikacja za pomocą modelu sieci neuronowej

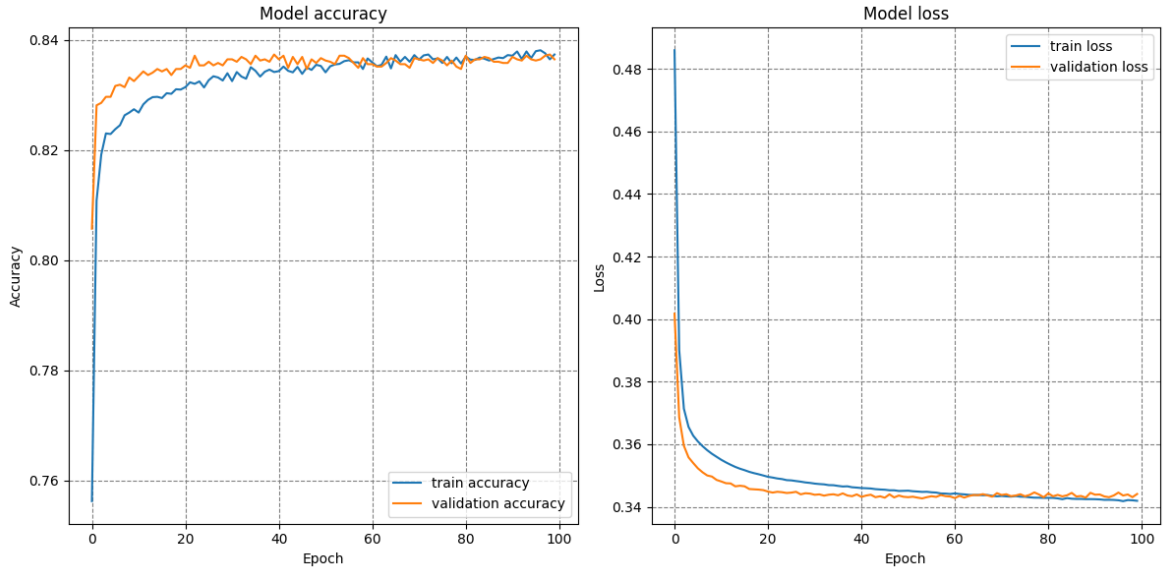
W tym przypadku zrobiłem klasyfikację za pomocą samodzielnie skonfigurowanego i wytrenowanego modelu sieci neuronowej. Udało mi się uzyskać dokładność klasyfikacji na poziomie: 83.56%.



Rysunek 5: Macierz błędu dla klasykikacji za pomocą modelu sieci neuronowej wytrenowanego na 100 epokach

```python
y_array = np.array(y)
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

encoder = OneHotEncoder()
y_encoded = encoder.fit_transform(y_array.reshape(-1, 1)).toarray()

X_train, X_test, y_train, y_test = train_test_split(X_scaled, y_encoded, test_size=0.3, random_state=42)

model = Sequential([
    Dense(64, activation='relu', input_shape=(X_train.shape[1],)),
    Dense(64, activation='relu'),
    Dense(y_encoded.shape[1], activation='softmax')
])

model.compile(optimizer='sgd', loss='binary_crossentropy', metrics=['accuracy'])

history = model.fit(X_train, y_train, epochs=100, validation_split=0.2)

test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=0)
print(f"Test Accuracy: {test_accuracy*100:.2f}%")
```

Powyżej kod, którego użyłem do wytrenowania i skonfigurowania modelu sieci neuronowej.



Powyższy wykres, przedstawia krzywe uczenia. Możemy odczytać z niego dokładność (accuracy) oraz stratę (loss) modelu podczas treningu i walidacji na 100 epokach.

5 Podsumowanie

Najlepszą dokładność uzyskałem klasyfikując dane za pomocą samodzielnie wytrenowanego na 100 epokach, modelu sieci neuronowej. Dokładność wyniosła wówczas 83.56%. Oczywiście, pewnie można uzyskać lepszą dokładność, nawet stosując te same funkcje aktywacji, optymalizatory i funkcje straty, lecz wymaga to również odrobiny szczęścia.