Dokumentacja Lab3

Wstępna analiza danych

Informacje o danych:

- Liczba wierszy: 4739
- Liczba kolumn: 15

Kolumny zawierają dane numeryczne oraz kategoryczne:

- Numeryczne: rownames (identyfikator), score (wynik do przewidzenia), unemp, wage, distance, tuition, education
- Kategoryczne: gender, ethnicity, fcollege, mcollege, home, urban, income, region

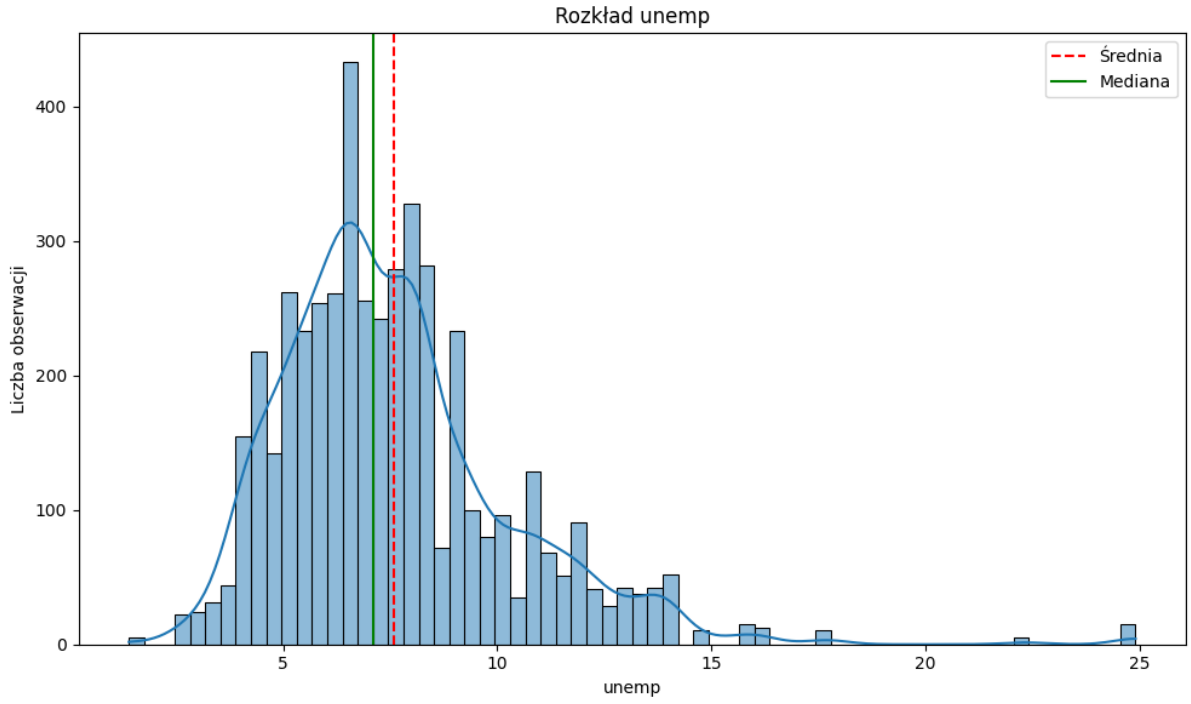
Brakujące dane:

- Wszystkie kolumny mają pełny zbiór danych, nie ma brakujących wartości.
- Brak danych może być reprezentowany poprzez wartości 0 w kolumnach numerycznych.

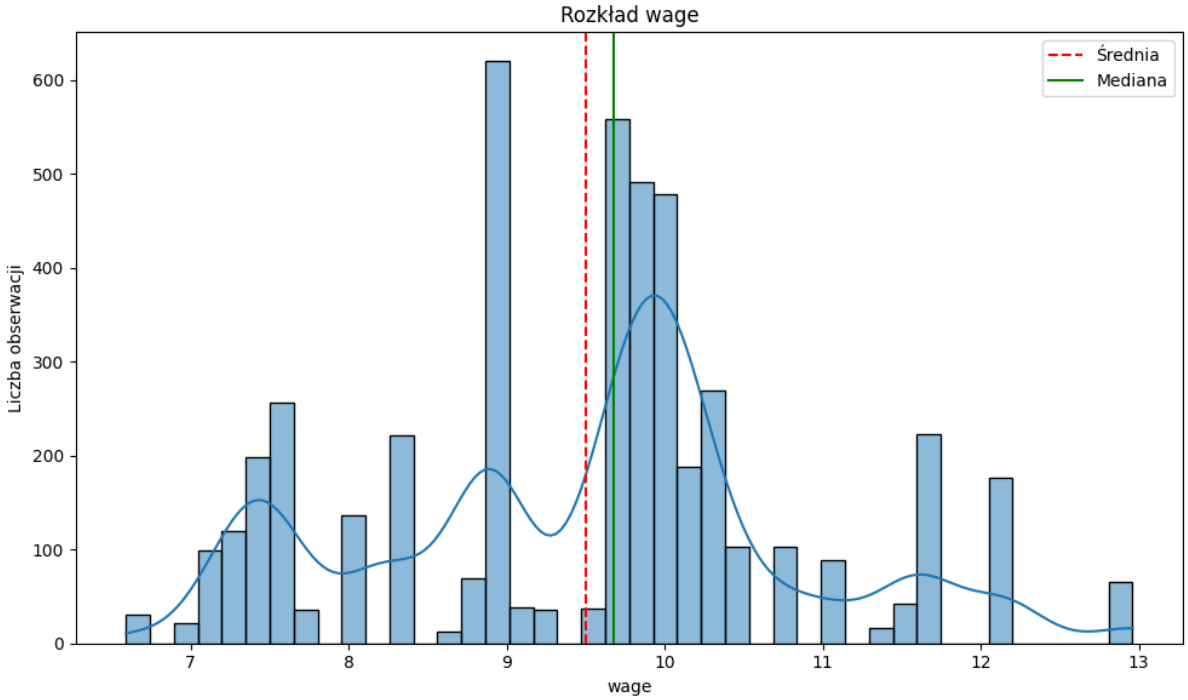
Analiza statystyczna:

Zmienne numeryczne:

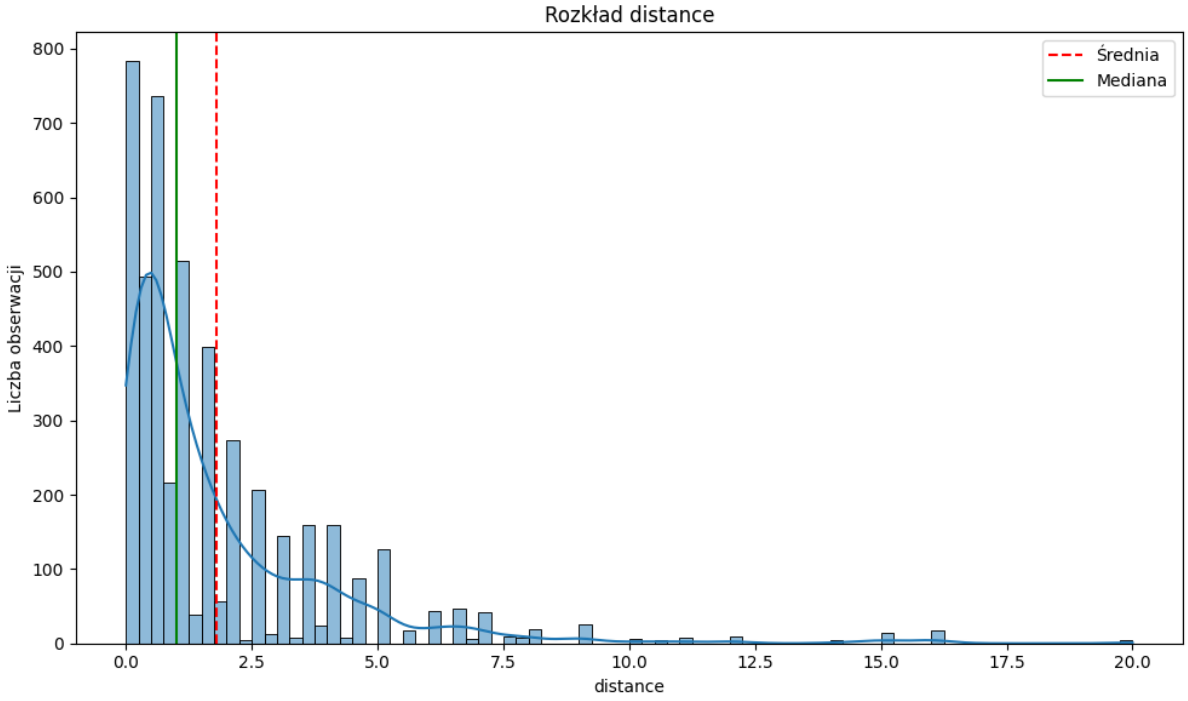
1. Unemployment (unemp):
   - Średnia: 7.6
   - Mediana: 7.1
   - Zakres wartości: 1.4 - 24.9



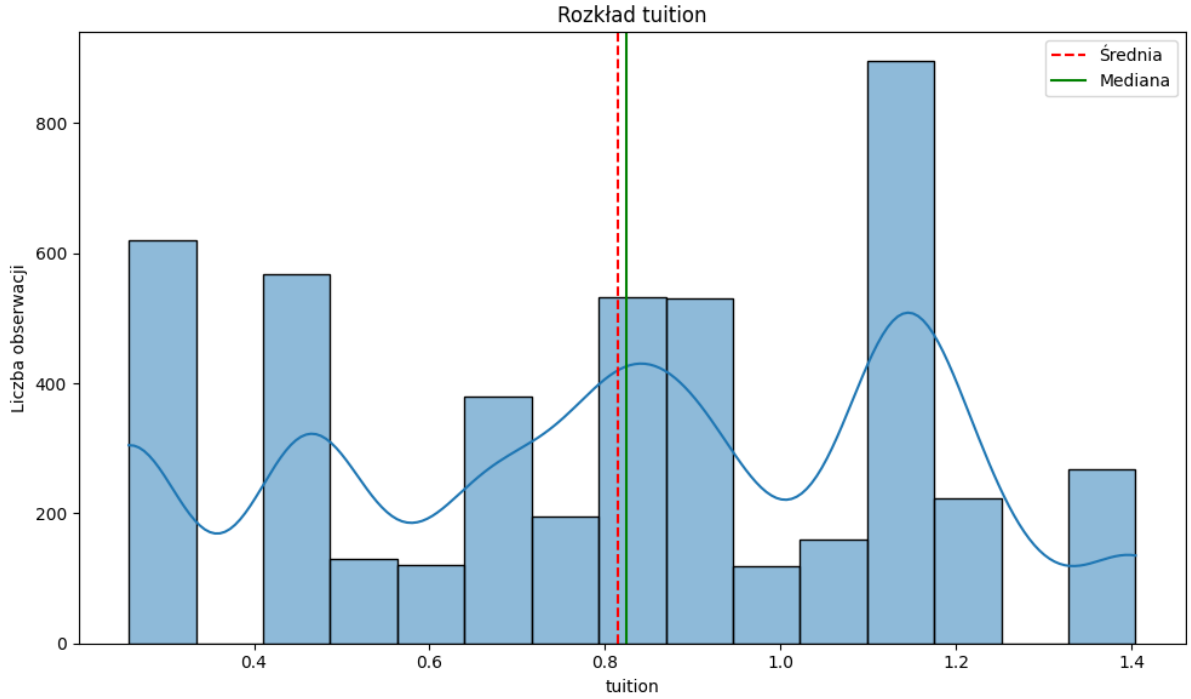
2. Wynagrodzenie (wage):
    - Średnia: 9.5
    - Mediana: 9.7
    - Zakres: 6.59 - 12.96



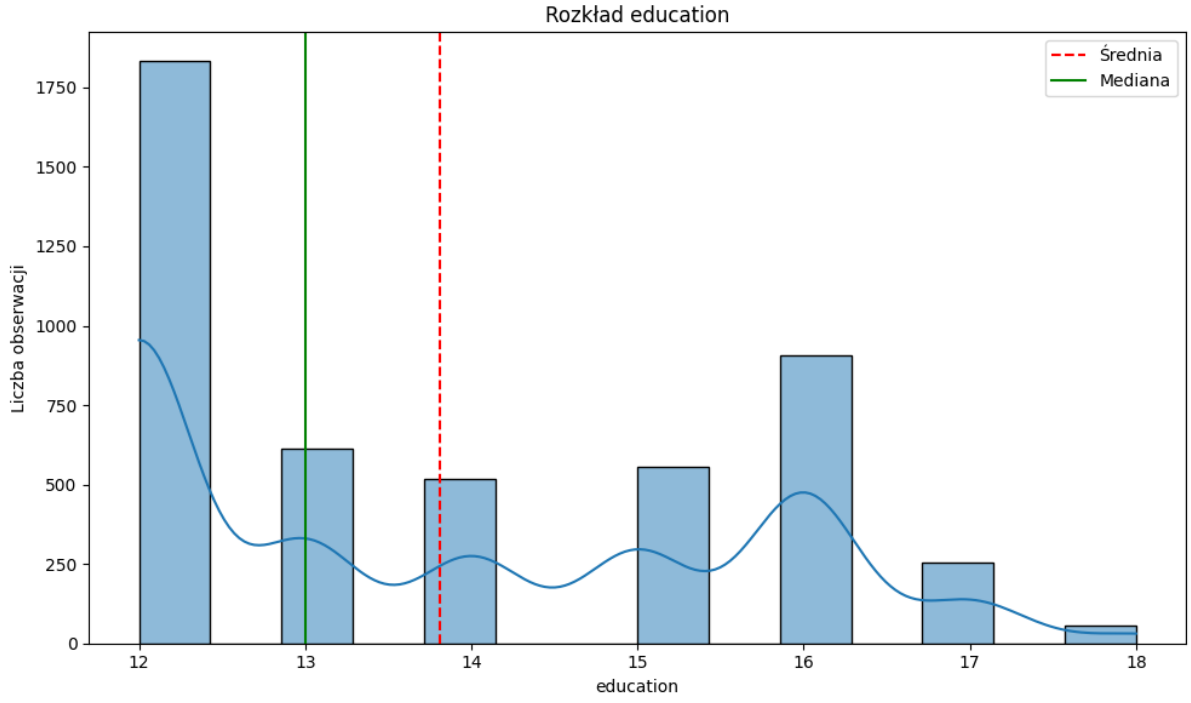
3. Dystans do uczelni (distance):
    - Średnia: 1.8
    - Mediana: 1.0
    - Zakres: 0.0 - 20.0



4. Czesne (tuition):
    - Średnia: 0.81
    - Mediana: 0.82
    - Zakres: 0.26 - 1.40

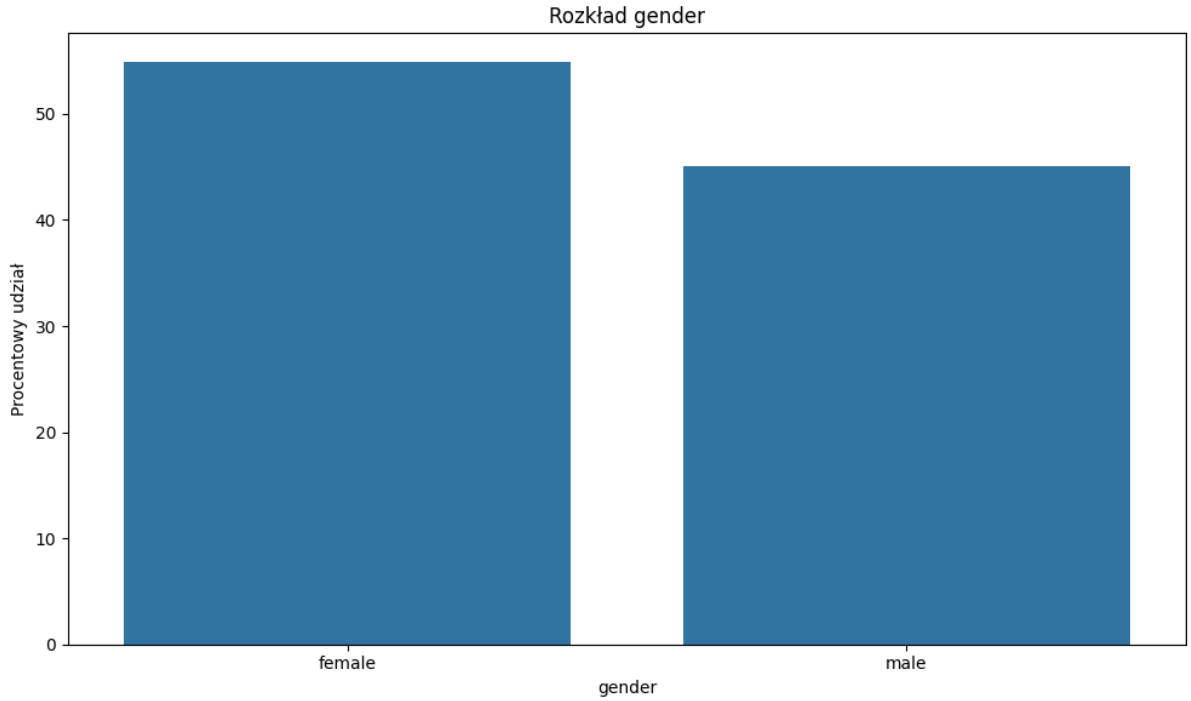


5. Edukacja (education):
    - Średnia: 13.8
    - Mediana: 13
    - Zakres: 12 - 18

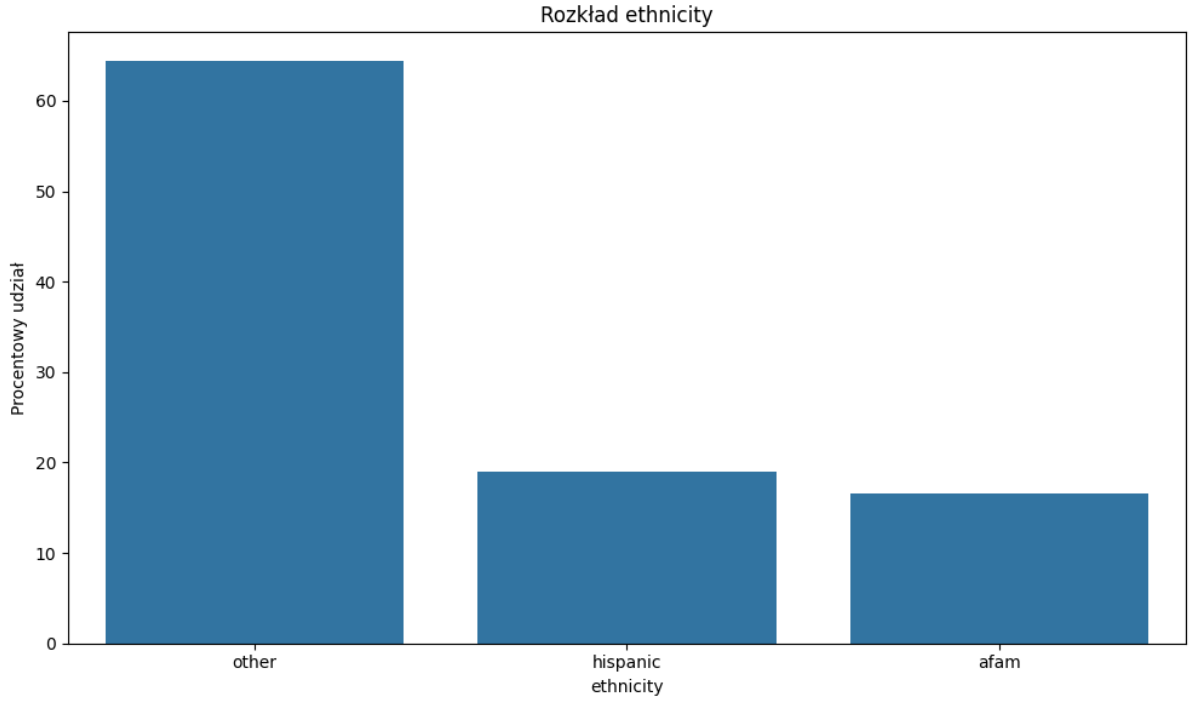


Zmienne kategoryczne:

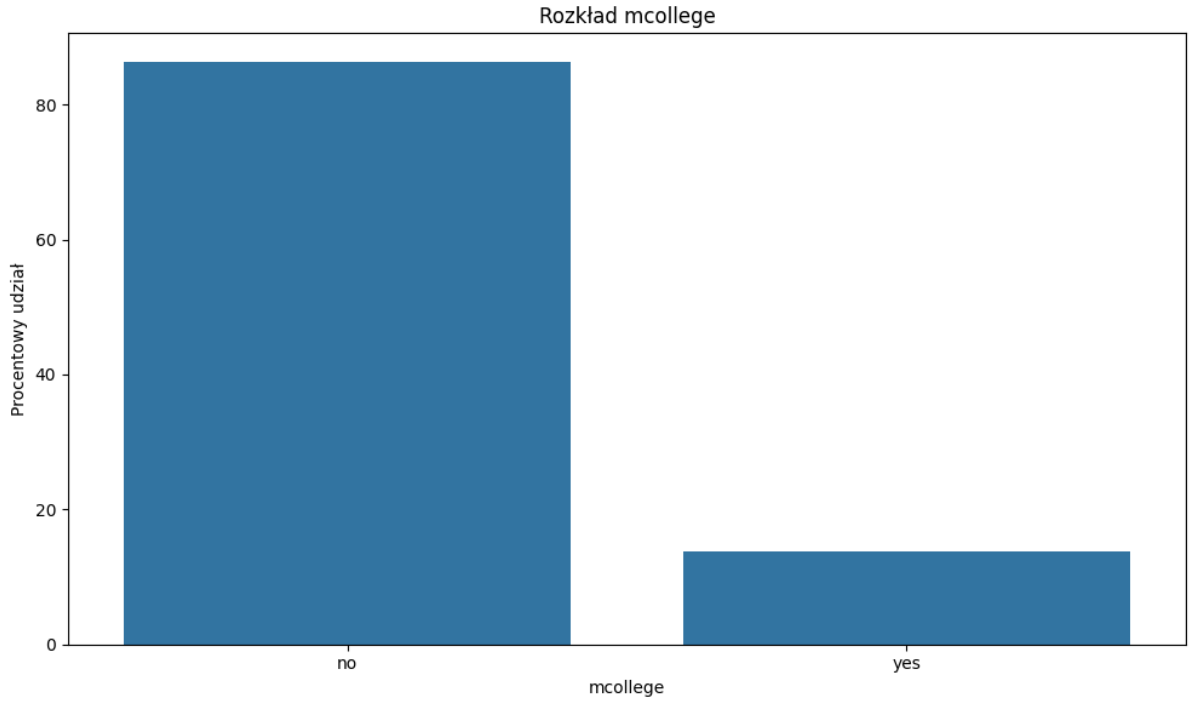
1. Płeć (gender):
   - Kategoria: male, female

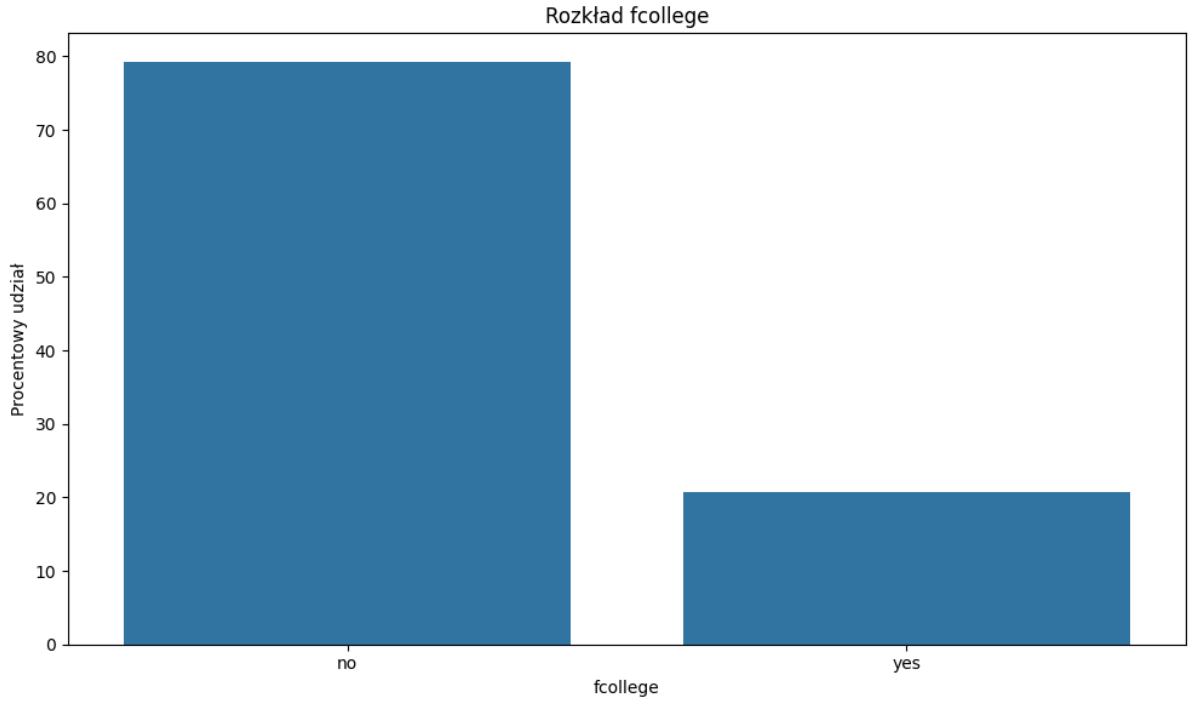


2. Etniczność (ethnicity):
   - Kategoria: afam (Afroamerykanin), hispanic, other



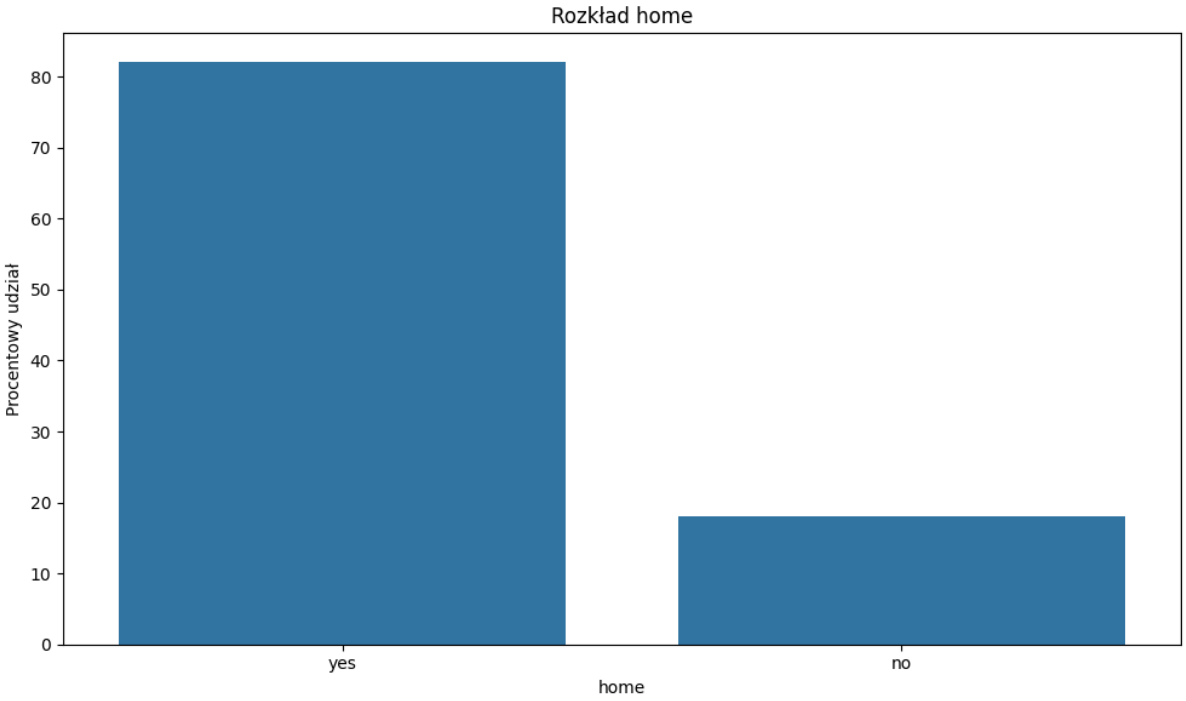
3. Rodzic ukończył studia (fcollege, mcollege):
   - Kategoria: yes, no





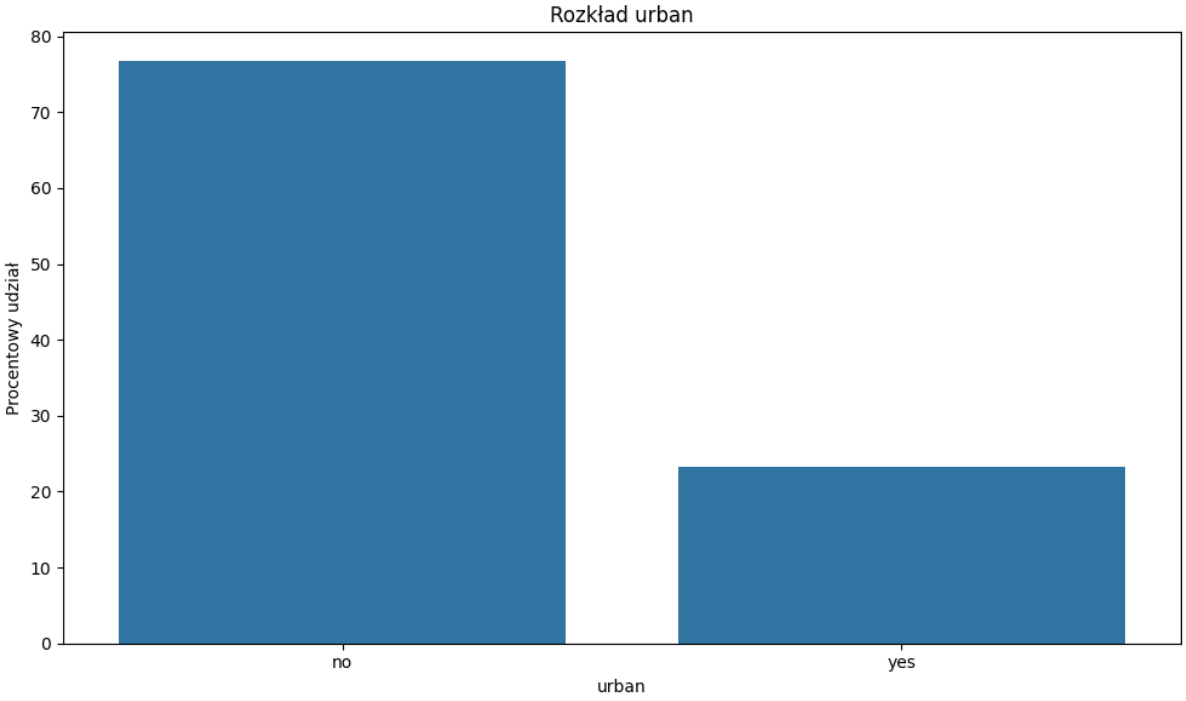
4. Dom (home):

- Kategoria: yes, no



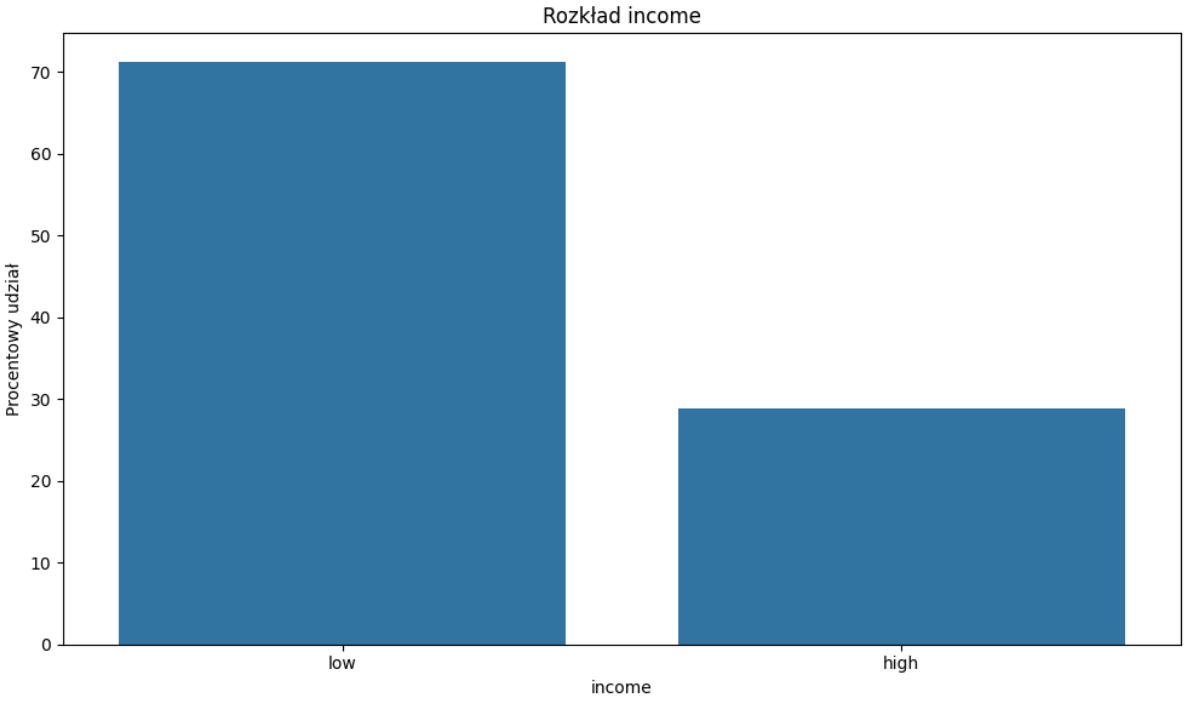
5. Obszar miejski (urban):

- Kategoria: yes, no



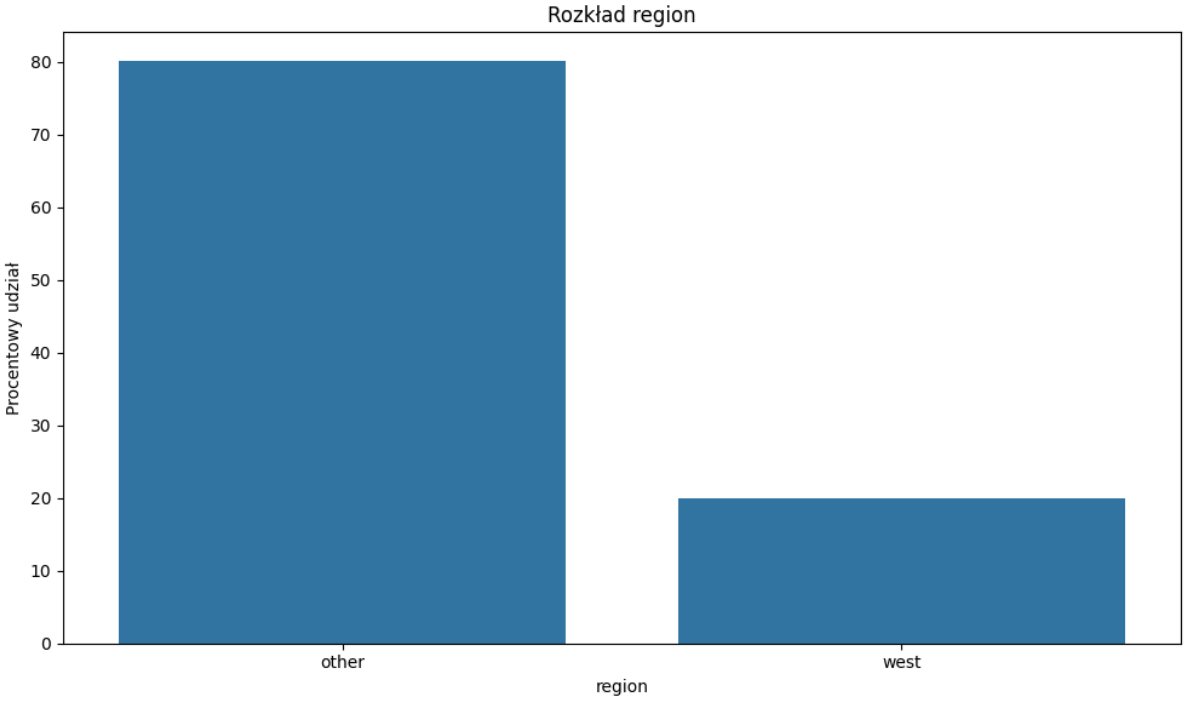
6. Dochód rodziny (income):

- Kategoria: high, low



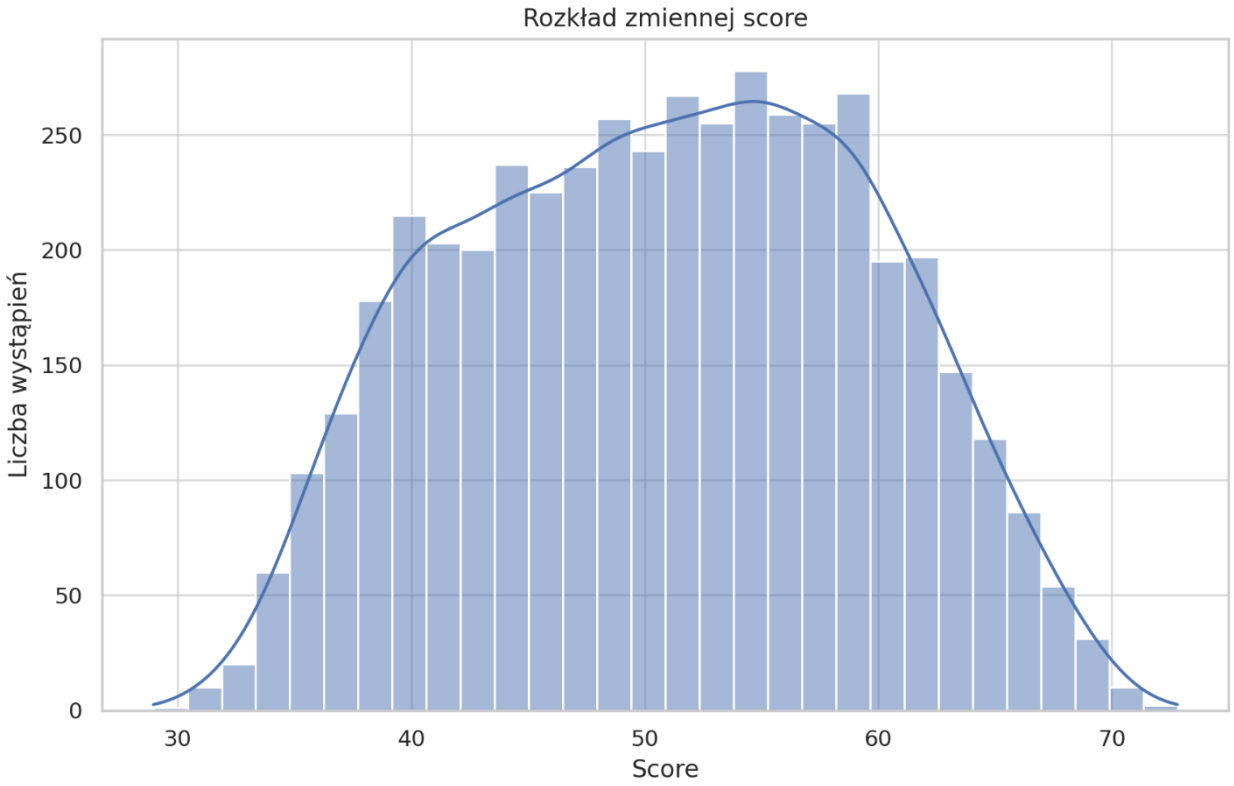
7. Region (region):

- Kategoria: west, other

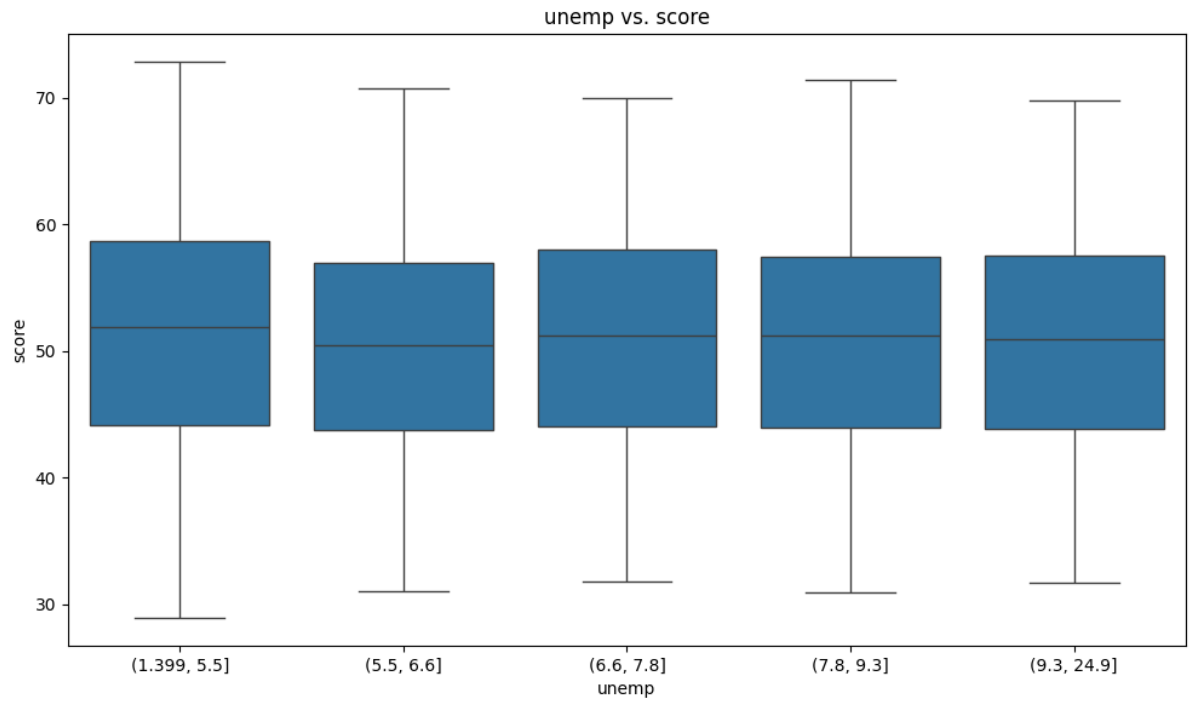


Wizualizacja zależności danych

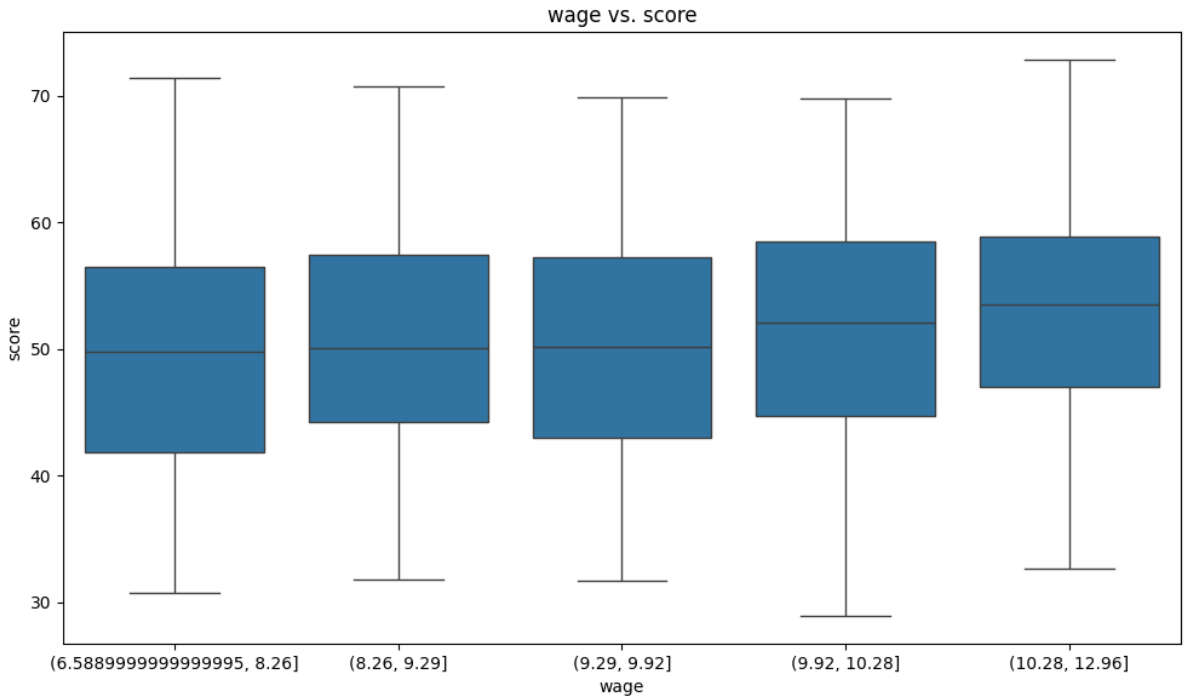
Rozkład zmiennej score jest zbliżony do normalnego, jednak istnieją pewne asymetrie i wartości odstające. Większość wyników znajduje się w przedziale od około 40 do 60.



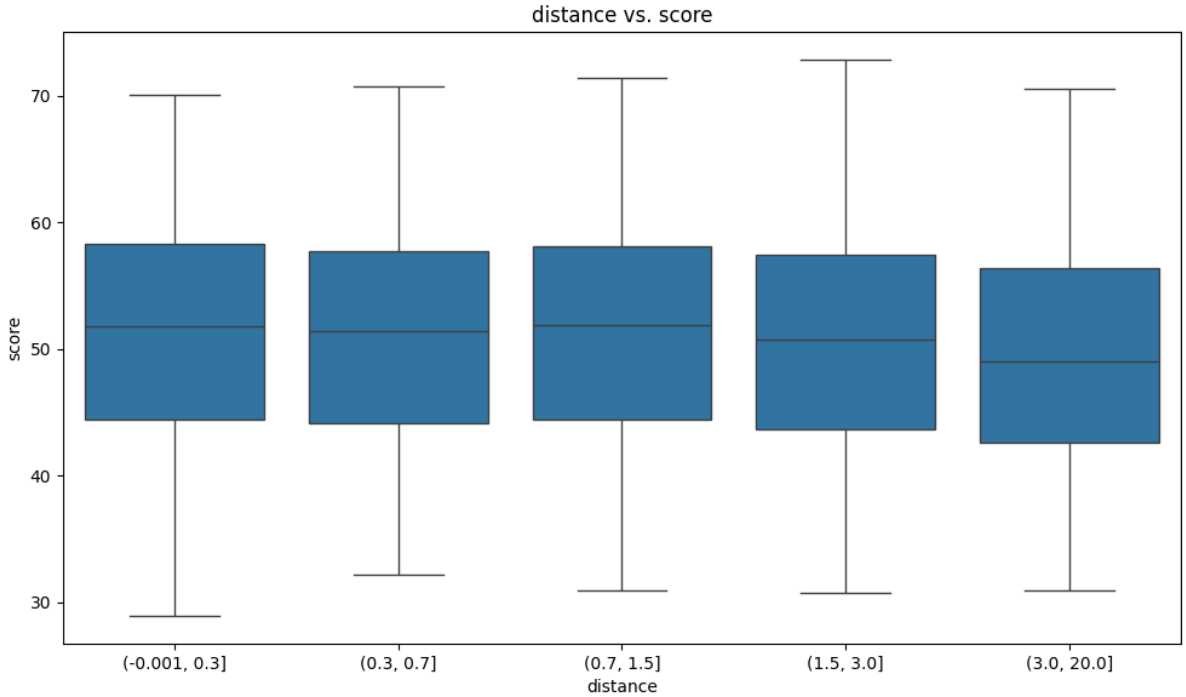
Grupy dla poziomów bezrobocia pokazują, że wyższy poziom bezrobocia nie ma jednoznacznej korelacji ze wzrostem lub spadkiem wartości zmiennej score.



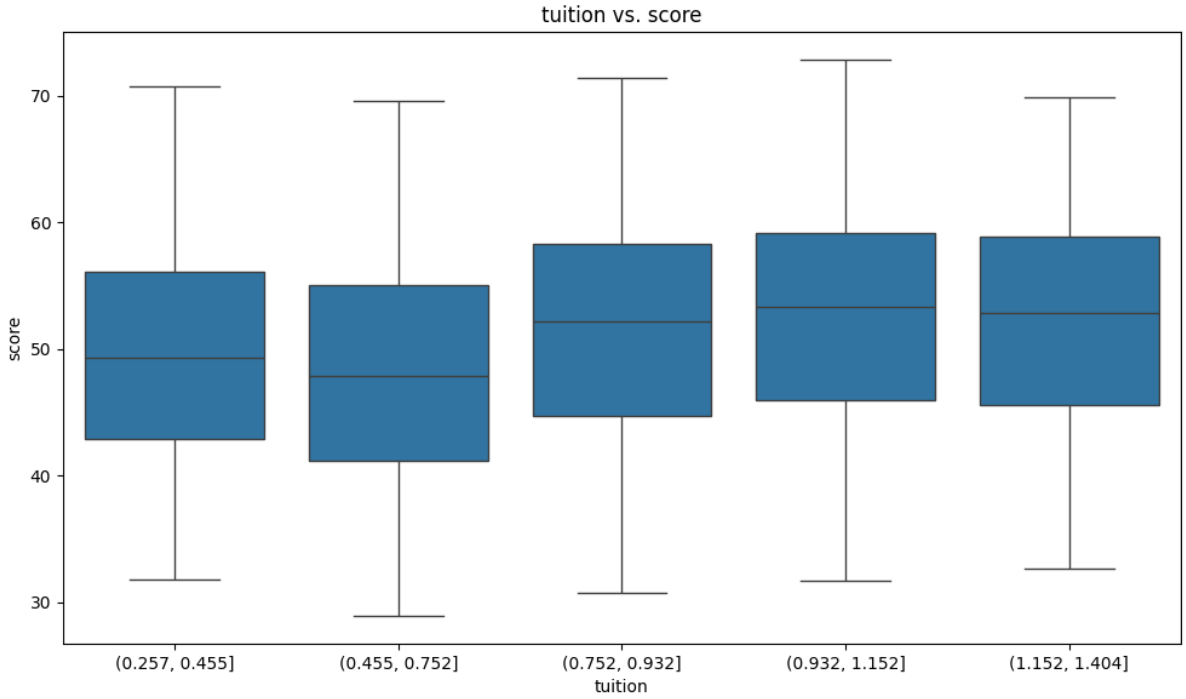
Wyższe wynagrodzenie ma umiarkowaną zależność z wyższymi wynikami score, jednak zależność nie jest linearna.



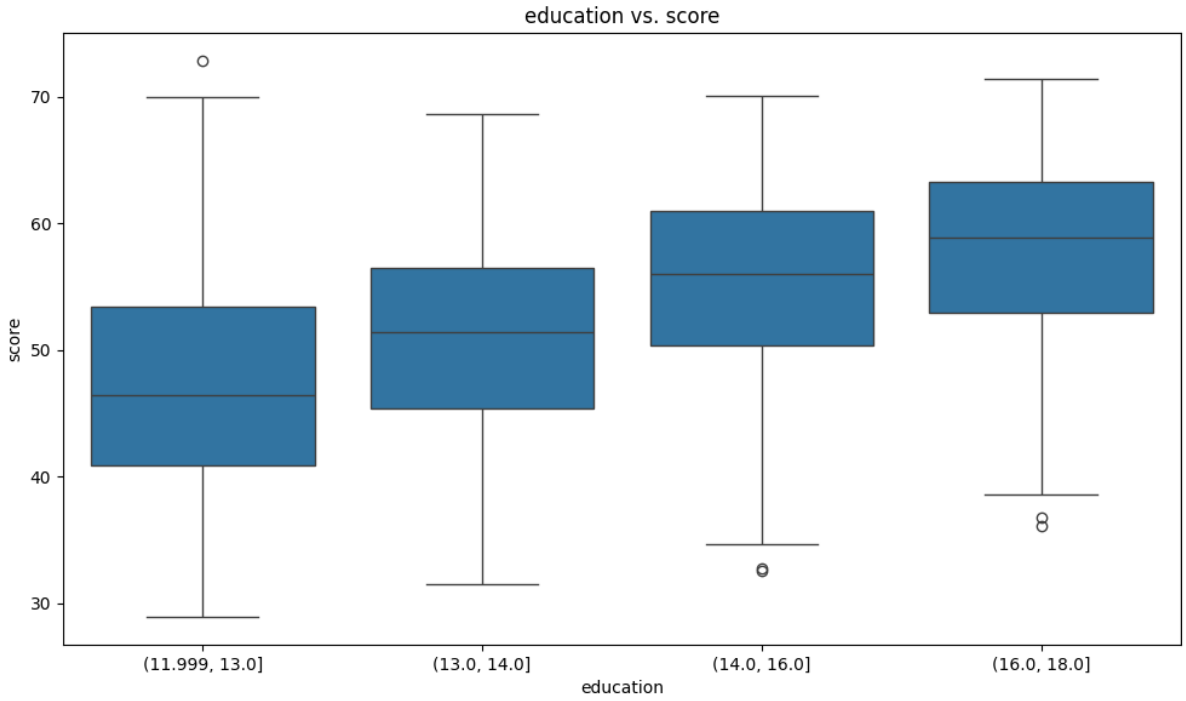
Dłuższy dystans ma tendencję do obniżenia wyniku, jednak w grupach powyżej 2.5 km wynik staje się bardziej zróżnicowany.



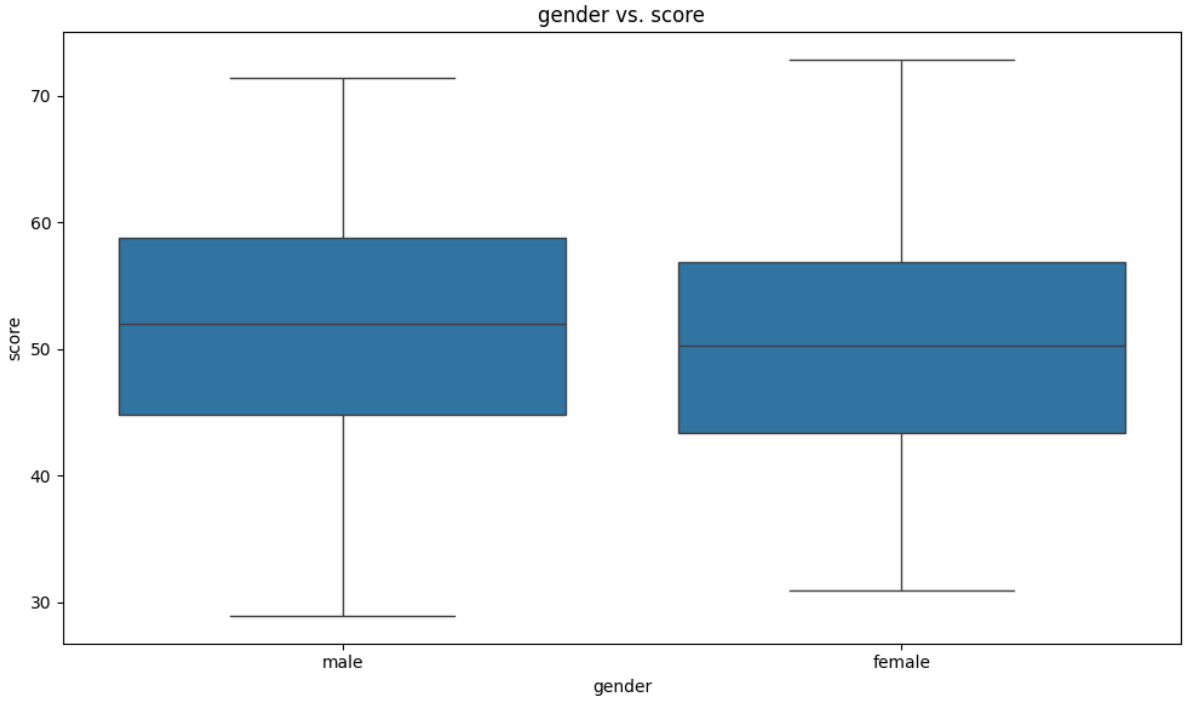
Wyższe czesne koreluje z nieznacznie wyższym wynikiem score.



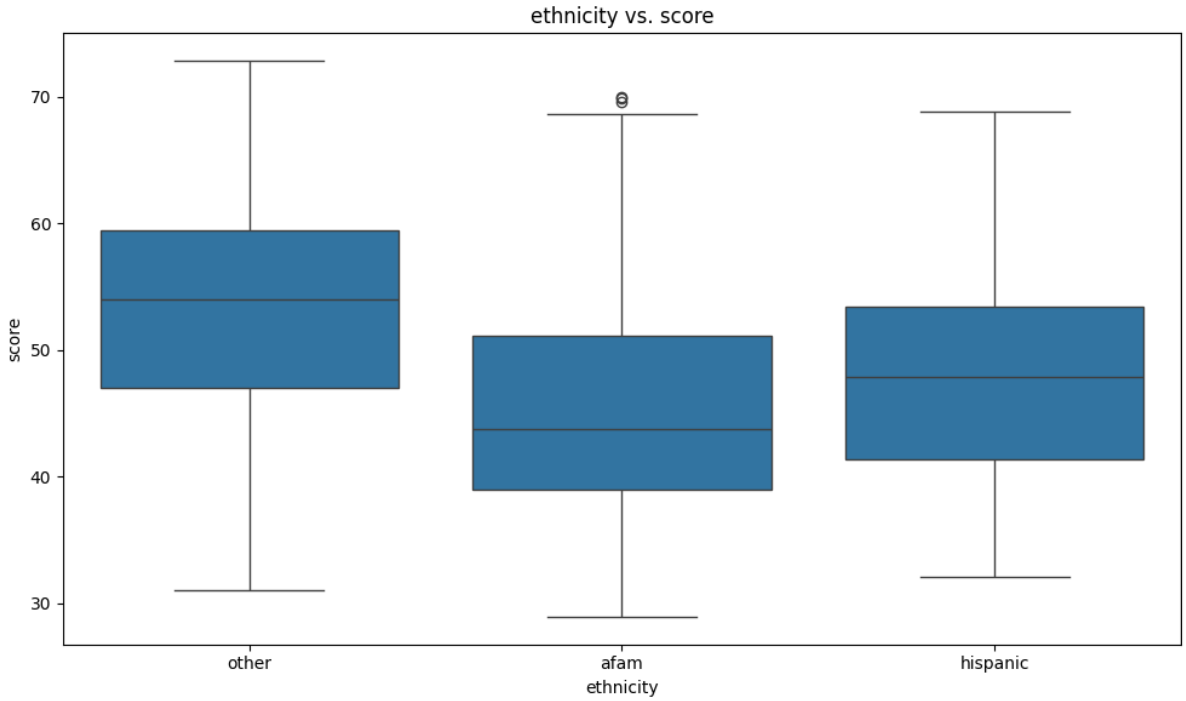
Większa wartość edukacji pokazuje trend w kierunku wyższych wyników score.



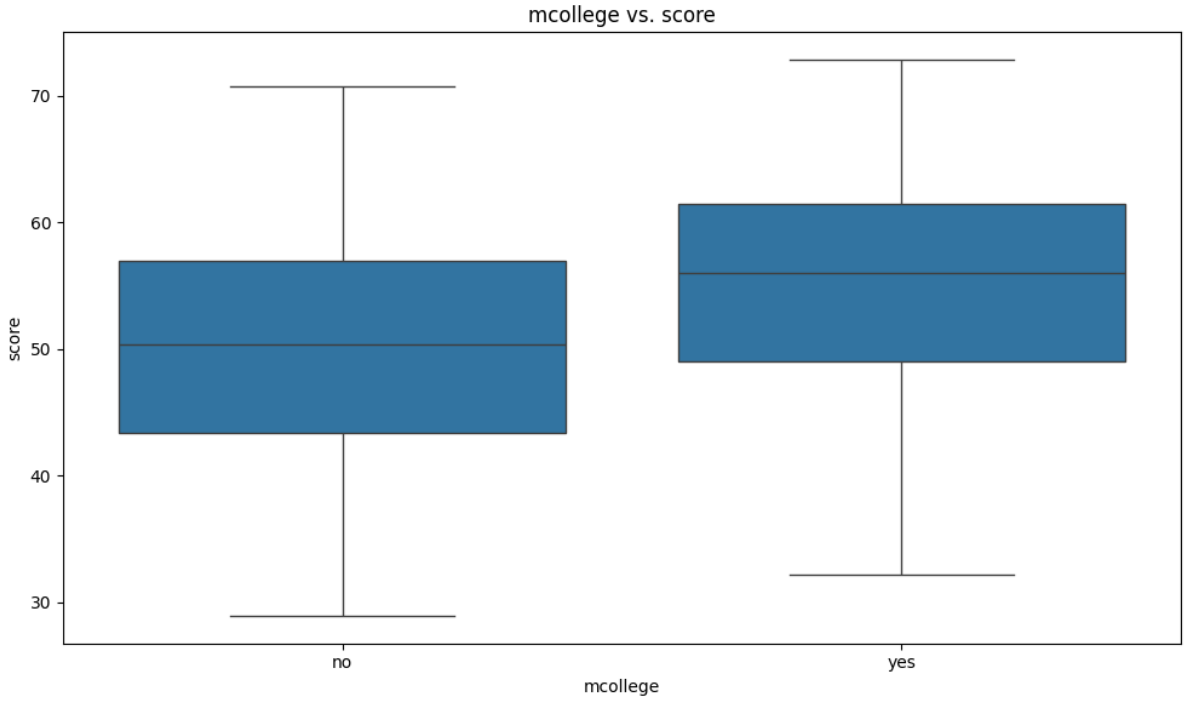
Kobiety mają nieco wyższe wyniki score w porównaniu do mężczyzn.

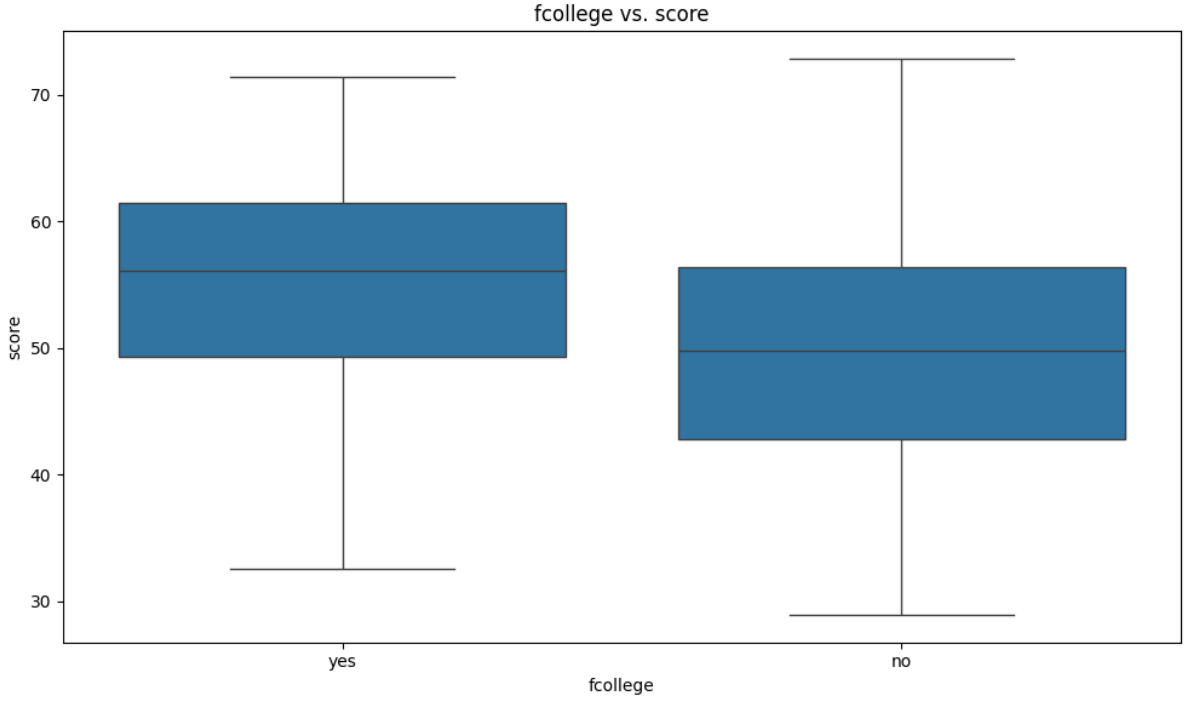


Różnice etniczne mają wpływ na wynik score, Afroamerykanie mają tendencję do osiągania niższych wyników.

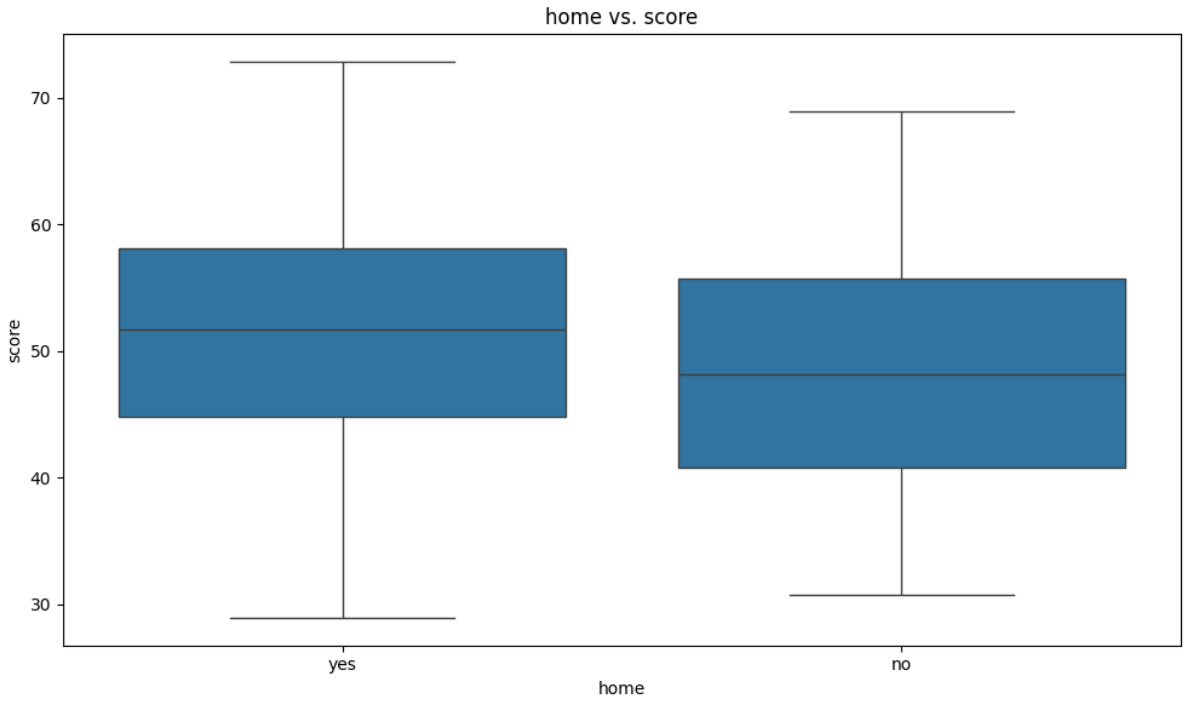


Rodzice z wykształceniem wyższym (zarówno matka, jak i ojciec) mają pozytywną korelację z wyższym wynikiem score.

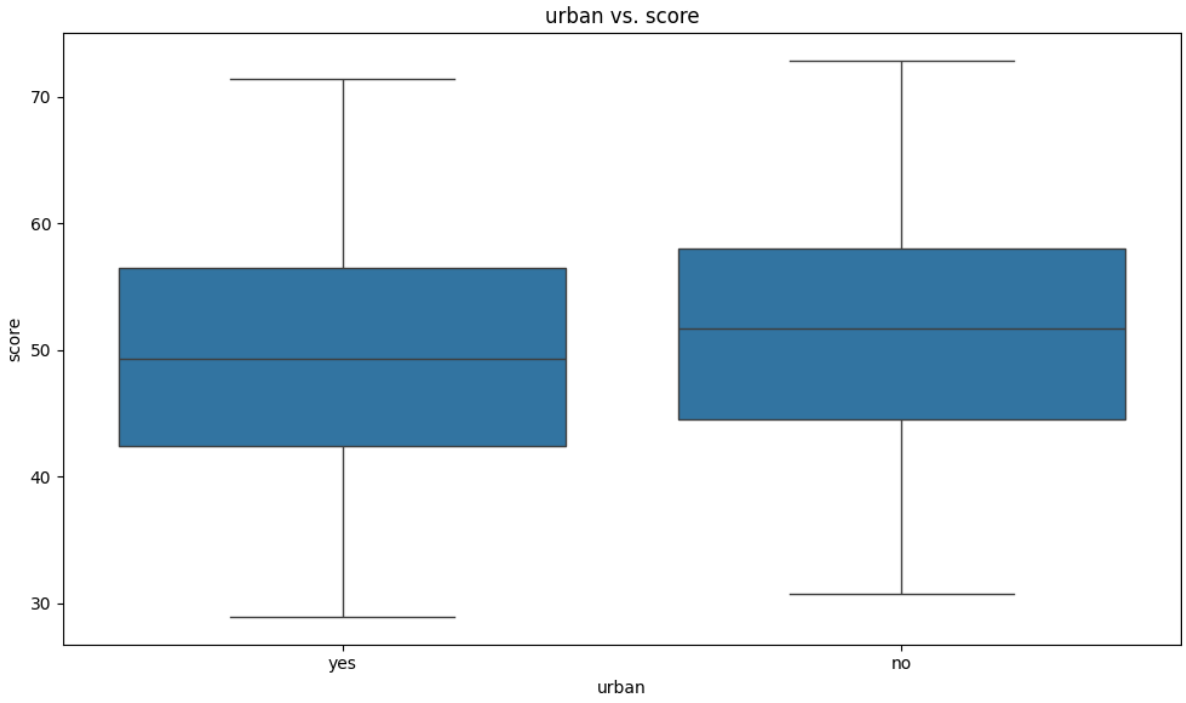




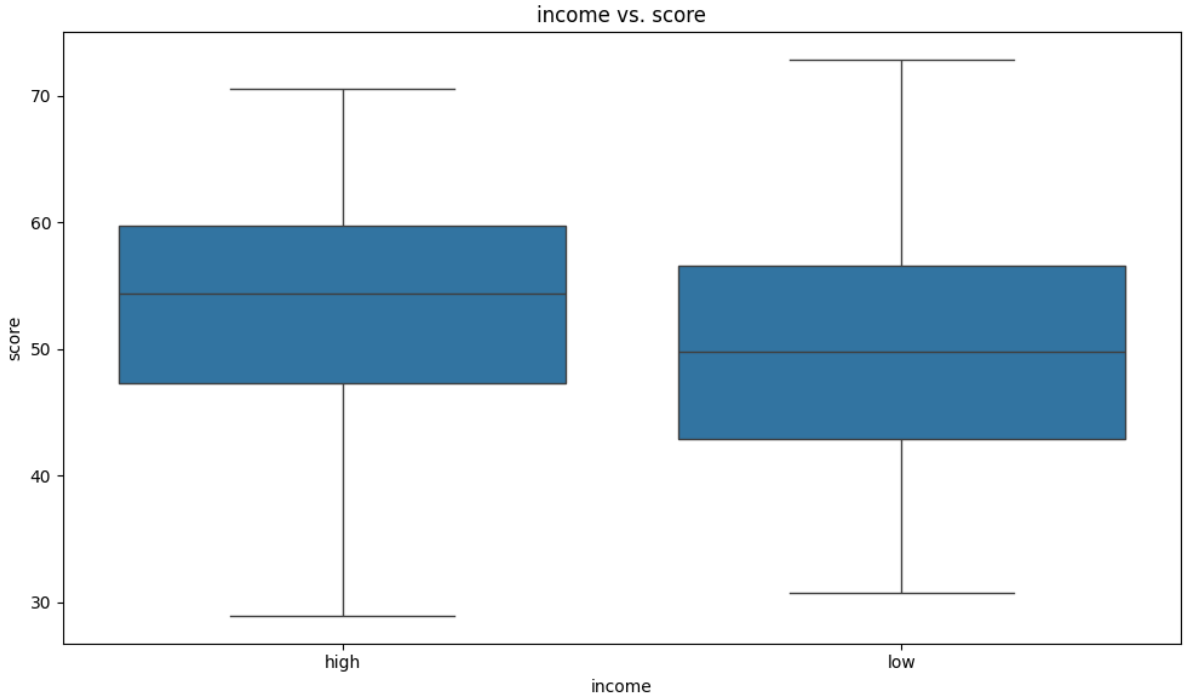
Studenci mieszkający w domu mają tendencję do niższych wyników score.



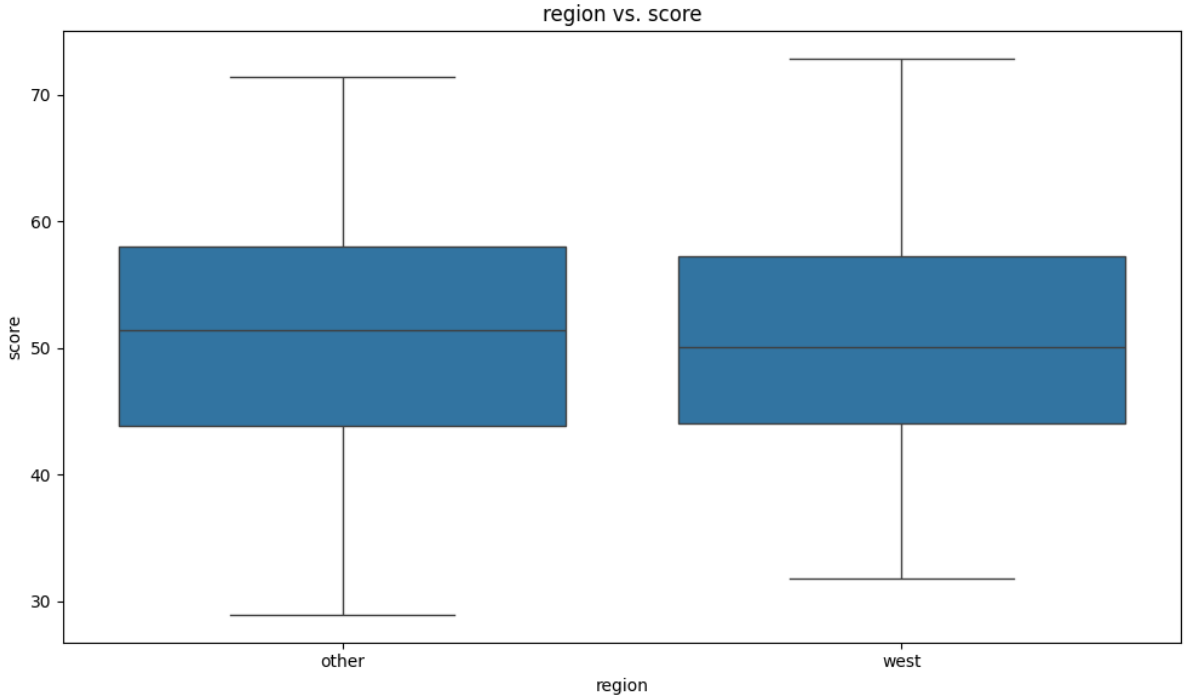
Mieszkanie w obszarze miejskim nieznacznie wpływa na wyższy wynik.



Wyższy dochód koreluje z wyższym wynikiem score.



Region ma wpływ na wyniki score, region "west" pokazuje wyższe wyniki w porównaniu do "other".



Inżynieria cech i przygotowanie danych

Podjęto następujące kroki, aby przygotować dane to pracy z modelem:

Imputacja

- W zmiennych numerycznych brakujące wartości są uzupełniane średnią (SimpleImputer(strategy='mean')).
- W zmiennych kategorycznych brakujące wartości są uzupełniane najczęściej występującą kategorią (SimpleImputer(strategy='most_frequent')).

Standaryzacja:

- Dla zmiennych numerycznych używana jest standaryzacja za pomocą StandardScaler. Każda zmienna jest przekształcana tak, aby miała średnią 0 i odchylenie standardowe 1.

One-Hot Encoding:

- Dla zmiennych kategorycznych używana jest technika one-hot encoding, która przekształca kategorie w zmienne binarne.

Podział danych:

- Dane są podzielone na zbiór treningowy i testowy w proporcji 80% na trening i 20% na test.

Kod odpowiedzialny za inżynierie cech i przygotowanie danych – data_prediction.py

```
# Wczytanie datasetu
df = pd.read_csv('./CollegeDistance.csv')

# Podział zmiennych
num_cols = ['unemp', 'wage', 'distance', 'tuition', 'education']
cat_cols = ['gender', 'ethnicity', 'fcollege', 'mcollege', 'home', 'urban', 'income', 'region']

# Zmienna przewidywana (target)
target = 'score'

# Tworzymy pipeline dla zmiennych numerycznych i kategorycznych
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='mean')),  # Imputacja wartości brakujących
    ('scaler', StandardScaler())  # Standaryzacja
])

categorical_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='most_frequent')),  # Imputacja najczęściej występujących wartości
    ('onehot', OneHotEncoder(handle_unknown='ignore'))  # One-hot encoding
])

# Połączenie transformacji dla zmiennych numerycznych i kategorycznych
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, num_cols),
        ('cat', categorical_transformer, cat_cols)
    ])

# Przygotowanie danych do modelu
X = df.drop(columns=[target])  # Zbiór cech (bez zmiennej score)
y = df[target]  # Zmienna przewidywana

# Podział danych na zbiór treningowy i testowy
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Wybór modelu AI

Zadanie polega na przewidywaniu zmiennej ciągłej score, co wskazuje na regresję, dlatego należy wybrać odpowiedni model regresyjny.

Potencjalne modele do rozważenia:

Regresja liniowa:

- Dobrze działa na stosunkowo prostych danych, gdzie zmienność przewidywanej zmiennej dobrze opisuje jedna lub więcej zmiennych wejściowych w liniowy sposób.
- Szybki i wydajny model do treningu, jednak zakłada liniową zależność, co może ograniczać jego skuteczność przy bardziej skomplikowanych zależnościach.

Lasy losowe (Random Forest):

- Model oparty na drzewach decyzyjnych, działający dobrze z bardziej złożonymi relacjami między zmiennymi oraz w przypadku danych, które nie są idealnie liniowe.
- Odporny na nadmierne dopasowanie (overfitting) dzięki technice baggingu.

Gradient Boosting (XGBoost, LightGBM):

- Zaawansowana technika regresji, skuteczna w modelowaniu skomplikowanych zależności. Wymaga większej mocy obliczeniowej, ale może dawać bardzo dobre wyniki, zwłaszcza na danych z nieliniowymi zależnościami.

Regresja LASSO / Ridge:

- Modele liniowe z regularyzacją. Regularyzacja pozwala na kontrolę nadmiernego dopasowania poprzez karanie modelu za zbyt duże współczynniki.

Random Forest

Random Forest będzie najlepszym modelem do rozwiązania tego problemu, ponieważ:

- Jest odporny na zmienne kategoryczne i dobrze radzi sobie z różnorodnością zmiennych.
- Może modelować złożone relacje między cechami, a regulacja liczby drzew i głębokości drzewa może zapobiec nadmiernemu dopasowaniu.
- W przeciwieństwie do regresji liniowej, nie wymaga założenia liniowości danych, co może być korzystne w przypadku tego zestawu danych, który zawiera zarówno zmienne numeryczne, jak i kategoryczne.

Trenowanie modelu

Po wytrenowaniu modelu stosując podstawowe ustawienia, uzyskano następujące wyniki:

- Wyniki dla zbioru treningowego: MSE = 10.4788, MAE = 2.5032, RMSE = 3.2371, R^2 = 0.8615, MAPE = 0.0513
- Wyniki dla zbioru testowego: MSE = 53.6624, MAE = 5.8536, RMSE = 7.3255, R^2 = 0.2924, MAPE = 0.1204

Widoczna jest duża dysproporcja pomiędzy zachowaniem modelu na danych treningowych, a testowych. Otrzymane wyniki nie są satysfakcjonujące i potrzebna jest optymalizacja modelu.

Wyjaśnienie metryk:

- Mean Squared Error (MSE): ocenia średni błąd kwadratowy pomiędzy przewidywaną a rzeczywistą wartością.

MAE (Mean Absolute Error):

- Średni błąd absolutny – mówi, jak bardzo w średniej model myli się przy przewidywaniu, bez kwadratowania różnic.

RMSE (Root Mean Squared Error):

- Pierwiastek z błędu średniokwadratowego – miara ta przedstawia błąd w jednostkach oryginalnej zmiennej, co może być łatwiej interpretowalne.

R-squared (R^2):

- Ocenia, jak dobrze nasz model dopasowuje się do danych.

MAPE (Mean Absolute Percentage Error):

- Średni błąd procentowy – miara przydatna, gdy chcemy zobaczyć, jak bardzo procentowo przewidywane wartości różnią się od rzeczywistych.

Kod odpowiedzialny za wytrenowanie modelu – data_prediction.py

```
80   #Trenowanie modelu Random Forest
81   rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
82
83   # Trenowanie modelu na zbiorze treningowym
84   rf_model.fit(X_train_prepared, y_train)
85
86   # Przewidywanie na zbiorze treningowym
87   y_train_pred = rf_model.predict(X_train_prepared)
88
89   # Przewidywanie na zbiorze testowym
90   y_test_pred = rf_model.predict(X_test_prepared)
91
92   # Ocena modelu - MSE, MAE, RMSE, R^2, MAPE
93   train_mse = mean_squared_error(y_train, y_train_pred)
94   train_mae = mean_absolute_error(y_train, y_train_pred)
95   train_rmse = np.sqrt(train_mse)
96   train_r2 = r2_score(y_train, y_train_pred)
97   train_mape = mean_absolute_percentage_error(y_train, y_train_pred)
98
99   test_mse = mean_squared_error(y_test, y_test_pred)
100  test_mae = mean_absolute_error(y_test, y_test_pred)
101  test_rmse = np.sqrt(test_mse)
102  test_r2 = r2_score(y_test, y_test_pred)
103  test_mape = mean_absolute_percentage_error(y_test, y_test_pred)
```

Optymalizacja modelu

Zdecydowanie warto przeprowadzić optymalizację modelu, ponieważ dotychczasowe wyniki na zbiorze testowym wskazują na nadmierne dopasowanie (overfitting) i słabe ogólne dopasowanie na nowych danych. Zastosowano następujące metody:

Walidacja krzyżowa (cross-validation)

- Walidacja krzyżowa pomaga uniknąć nadmiernego dopasowania, dzieląc dane na kilka różnych podzbiorów. Na każdym podzbiorze model jest trenowany i oceniany, co daje bardziej wiarygodną ocenę modelu.

Tunowanie hiperparametrów (Grid Search)

- Do tuningu hiperparametrów użyta została metoda GridSearchCV (metoda przeszukiwania wszystkich możliwych kombinacji), aby znaleźć najlepsze parametry dla modelu Random Forest.

Po zastosowaniu optymalizacji modelu otrzymano następujące wyniki:

- Wyniki na zbiorze testowym: MSE = 49.3515, MAE = 5.7161, RMSE = 7.0251, R^2 = 0.3492, MAPE = 0.1180

Powyższe wyniki dalej znacząco różnią się od tych uzyskanych pierwotnie na zbiorze treningowym.

Kod odpowiedzialny za optymalizacje modelu – data_prediction.py

```python
# Rozpoczęcie optymalizacji na nowym modelu podstawowym
rf_base = RandomForestRegressor(random_state=42)

# Parametry do tuningu
param_grid = {
    'n_estimators': [100, 200, 300],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'bootstrap': [True, False]
}

# Użycie GridSearchCV do tuningu hiperparametrów
grid_search = GridSearchCV(estimator=rf_base, param_grid=param_grid,
                           cv=5, n_jobs=-1, verbose=2, scoring='neg_mean_squared_error')

# Dopasowanie modelu na danych treningowych
grid_search.fit(X_train_prepared, y_train)

# Najlepsze parametry
best_params = grid_search.best_params_

# Logowanie najlepszych parametrów
logging.info(f"Najlepsze parametry dla modelu Random Forest: {best_params}")

# Wytrenowanie modelu z najlepszymi parametrami
rf_best = grid_search.best_estimator_

# Ocena modelu na zbiorze testowym
y_test_pred_best = rf_best.predict(X_test_prepared)

# Obliczanie metryk dla zoptymalizowanego modelu
test_mse_best = mean_squared_error(y_test, y_test_pred_best)
test_mae_best = mean_absolute_error(y_test, y_test_pred_best)
test_rmse_best = np.sqrt(test_mse_best)
test_r2_best = r2_score(y_test, y_test_pred_best)
test_mape_best = mean_absolute_percentage_error(y_test, y_test_pred_best)
```

Możliwe przyczyny słabych wyników modelu

Skuteczność modelu Random Forest:

- Choć Random Forest jest potężnym algorytmem, czasem jego zdolność do generalizacji może być ograniczona, szczególnie w przypadku trudnych problemów regresyjnych. Może to wynikać z nieadekwatności cech do przewidywania zmiennej docelowej (score).

Ograniczona informacyjność cech:

- Cechy w zbiorze danych mogą nie być wystarczająco informatywne lub mogą zawierać zbyt wiele szumu, co utrudnia dokładne przewidywania.

Złożoność zależności:

- Relacje między zmiennymi a celem (score) mogą być nieliniowe lub zbyt skomplikowane, co powoduje, że Random Forest nie jest w stanie w pełni uchwycić tych zależności.

Alternatywne modele AI

W ramach testów przeprowadzono również testy modelu typu Gradient Boosting, a dokładniej XGBoost. Poniższe wyniki nadal nie były satysfakcjonujące i nie zostały umieszczone w projekcie z tego powodu.

- Zbiór testowy: MSE = 48.5271, MAE = 5.7489, RMSE = 6.9661, R^2 = 0.3601, MAPE = 0.1191