

Założenia projektowe

1. Skład zespołu:

- a. Marianna Tybura 259170
- b. Zuzanna Zając 259171
- c. Jakub Sońta 259166
- d. Krzysztof Kaniuka 259158

2. Cel: Predykcja wyników egzaminów studentów przy użyciu regresji

3. Zbiór danych: [Exam Score Prediction Dataset](#) (20 tys. Rekordów)

4. Technologia:

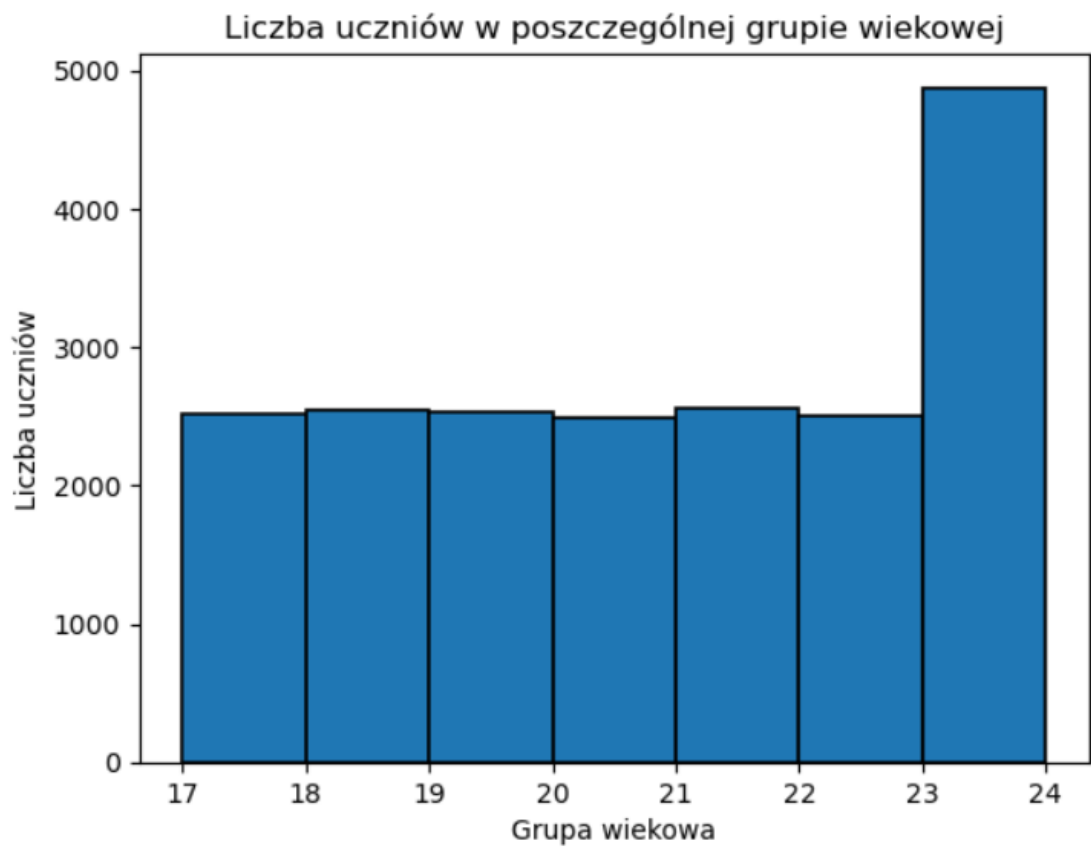
- a. Python
 - i. Pandas
 - ii. Matplotlib
 - iii. Seaborn
 - iv. Sklearn
 - v. Numpy

5. Model regresji: XGB Regressor

6. Cechy:

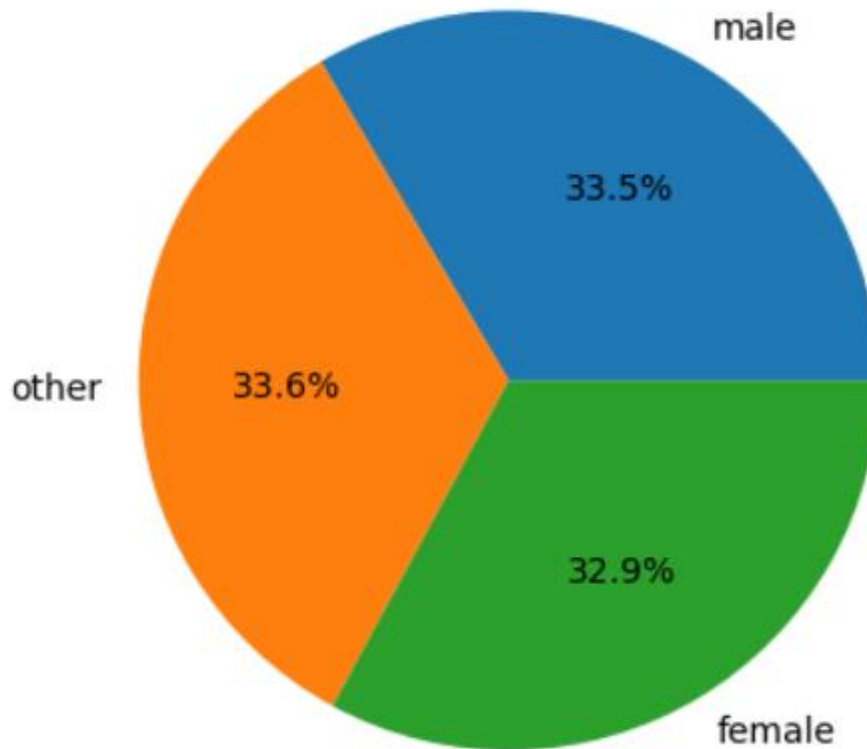
- 1. Age

Przedział wiekowy uczniów od 17 do 24 lat



2. Gender

Rozkład płciowy uczniów



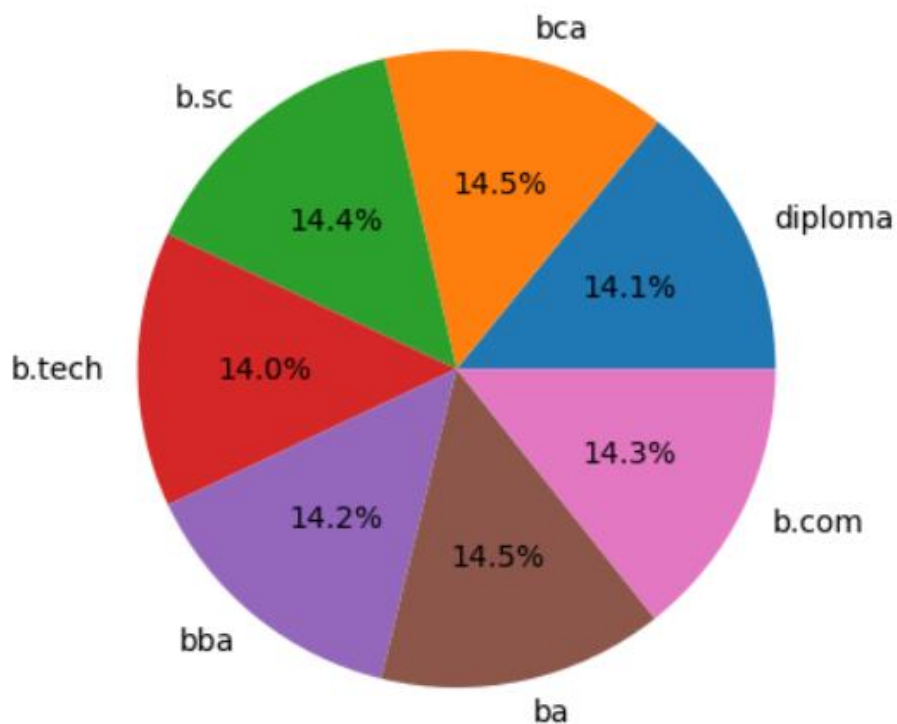
3. Course

Programy studiów, na które uczęszczali uczniowie

- Diploma - kwalifikacja po programie dyplomowym
- Bca - licencjat z zastosowań informatyki
- B.sc - licencjat nauk ścisłych/przyrodniczych
- B.tech - licencjat/inżynier w dziedzinie technologii
- Bba - licencjat z zarządzania
- Ba - licencjat nauk humanistycznych/społecznych
- B.com- licencjat z handlu/ekonomii

Poniżej widać że zbiór danych jest zbalansowany w kontekście tej cechy

Odsetek uczniów realizujących konkretne programy nauczania



4. Study hours

Liczba godzin poświęconych nauce (cecha numeryczna)

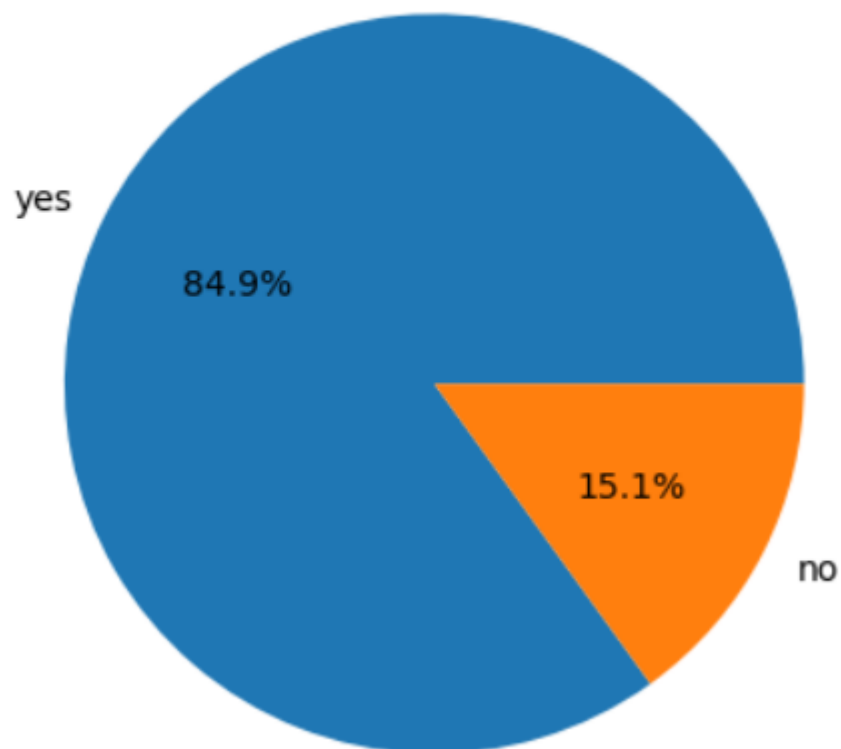
5. Class attendance

Obecność na zajęciach mierzona w skali od 1 do 100 (cecha numeryczna)

6. Internet access

Większość uczniów miała dostęp do Internetu

Dostęp do internetu



Cecha ta nie miała jednak dużego wpływu na rezultaty egzaminu

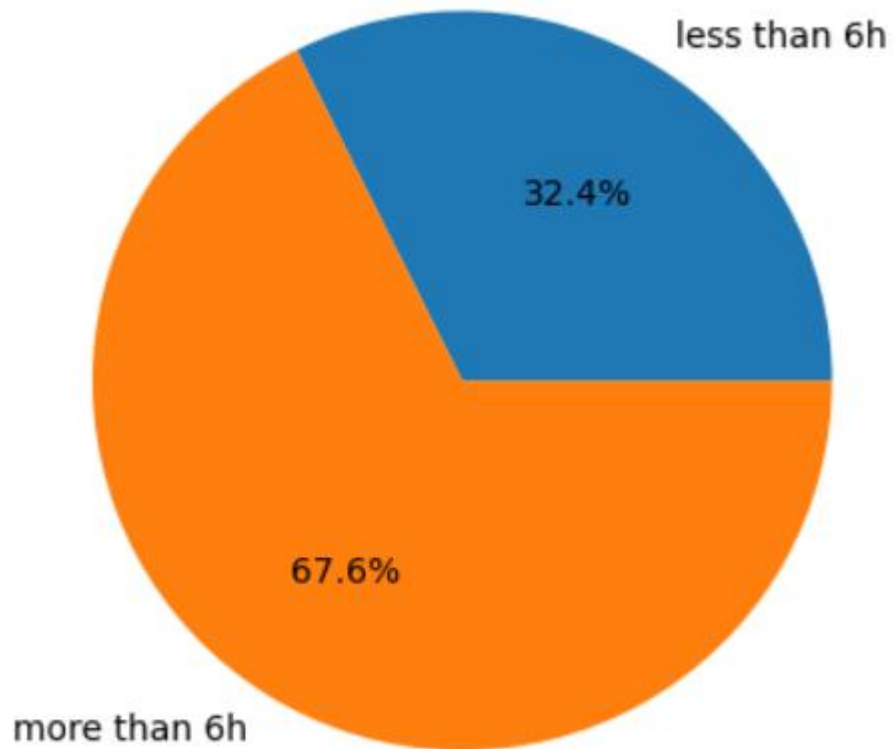
Mediana uczniów mających dostęp do Internetu 62.6

Mediana uczniów mających dostęp do Internetu 62.650000000000006

7. Sleep hours

Prawie 1/3 uczniów spała mniej niż 6 godzin

Czas snu uczniów



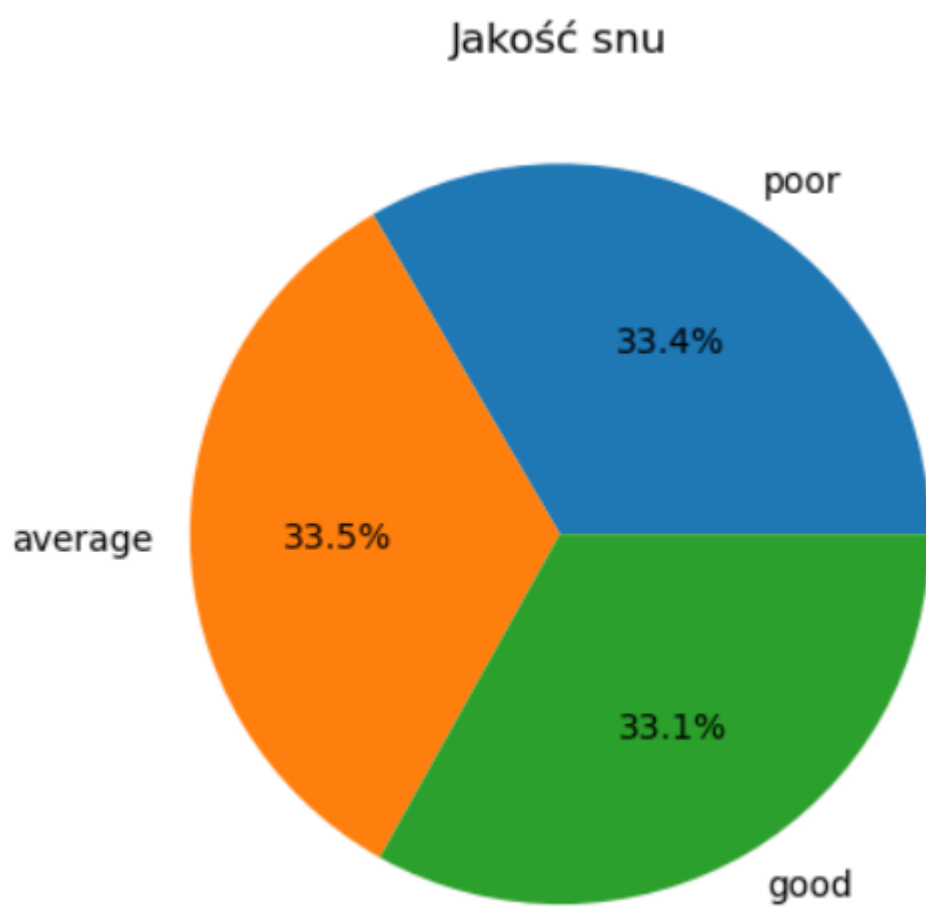
Długość snu ma większy wpływ na wynik egzaminu niż dostęp do Internetu

Mediana uczniów śpiących mniej niż 6 godzin 59.8

Mediana uczniów śpiących 6 lub więcej godzin 64.1

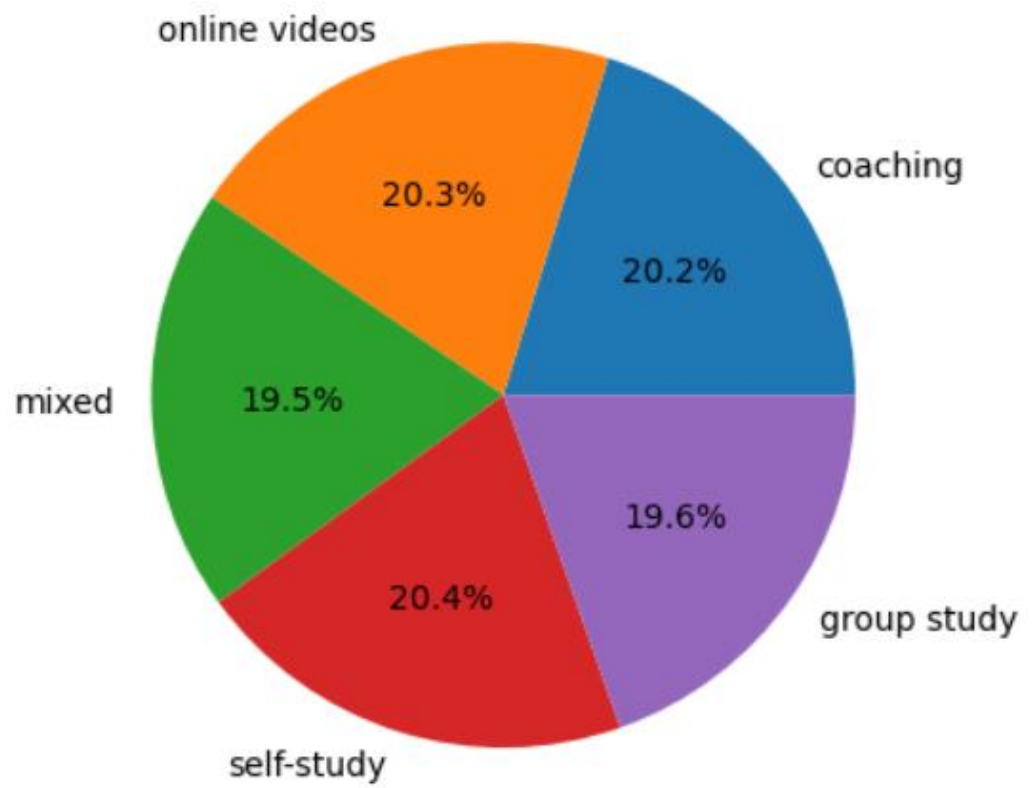
8. Sleep quality

Zbiór danych jest zbalansowany dla tej zmiennej



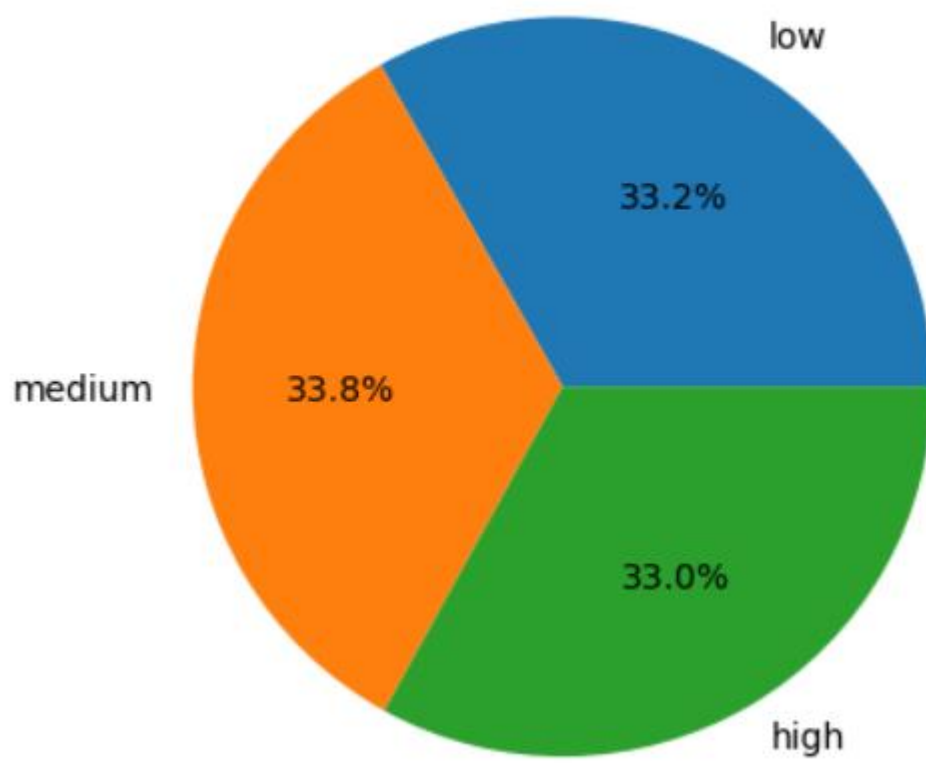
9. Study method

Sposób nauki



10. Facility Rating

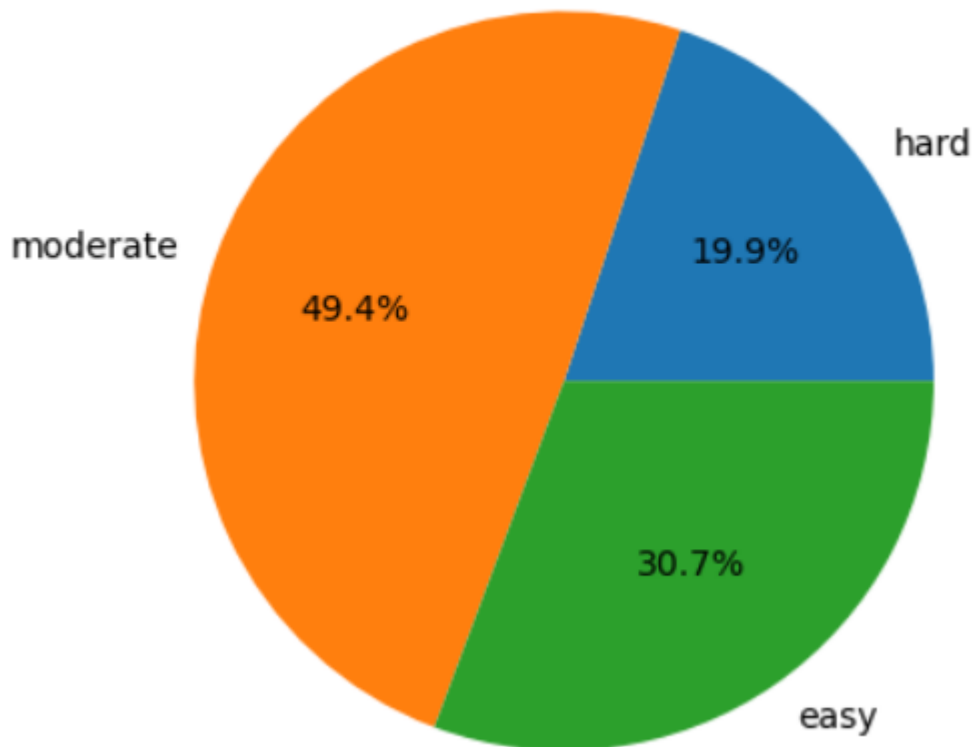
Warunki do nauki



11. Exam Difficulty

Niemal połowa egzaminów oceniana była jako średnio-trudna

Poziom trudności egzaminu



12. Exam score – zmienna zależna zbioru

Wynik egzaminu podany w skali od 0 do 100 (wartość numeryczna)

7. Omówienie modelu predykcyjnego

Pierwszym krokiem, po wczytaniu zbioru danych z pliku, było uzupełnienie braków danych. Dla wartości kategoriycznych dokonano tego wartością modalną, a dla wartości numerycznych, wartością średnią.

```
df_full = pd.read_csv("Exam_Score_Dataset.csv")  
  
for col in df_full.columns:  
    if df_full[col].dtype == 'object':  
        df_full[col] = df_full[col].fillna(df_full[col].mode()[0])  
    else:  
        df_full[col] = df_full[col].fillna(df_full[col].mean())
```

Następnie, usunięto ze zbioru kolumnę id, która nie niosła za sobą żadnej informacji w kontekście treningu modelu.

```
✓ if 'student_id' in df_full.columns:  
    df_full = df_full.drop(columns=['student_id'])
```

Zbiór danych podzielono na podzbiory treningowy i testowy. Ponadto wytypowano w zbiorze zmienną celu oraz wyróżniono wśród pozostałych cech bazowych cechy o charakterze numerycznym.

```
train_df = df_full.sample(frac=0.8, random_state=42).copy()  
test_df = df_full.drop(train_df.index).copy()  
  
train_df = train_df.reset_index(drop=True)  
test_df = test_df.reset_index(drop=True)  
  
TARGET = 'exam_score'  
base_features = [col for col in train_df.columns if col != TARGET and train_df[col].dtype == 'object']  
num_features = ['study_hours', 'class_attendance', 'sleep_hours']
```

W celu wstępnej obróbki podzbiorów danych, przygotowano funkcję `preproces(df)`.

W ramach tej funkcji, dokonano transformacji logarytmicznej oraz kwadratowej dla zmiennych numerycznych zbioru w celu zmniejszenia wpływu skośności rozkładu.

```
✓ in preprocess function  
✓ def preprocess(df):  
    df_temp = df.copy()  
  
    for col in num_features:  
        if col in df_temp.columns:  
            df_temp[f'log_{col}'] = np.log1p(df_temp[col])  
  
    for col in num_features:  
        if col in df_temp.columns:  
            df_temp[f'{col}_sq'] = df_temp[col] ** 2
```

Na podstawie obserwacji zbioru danych, dostrzeżono również potencjał zmiennych **study_hours**, **class_attendance** oraz **sleep_hours** na bazie których zbudowana została funkcja celu dla danego zbioru danych.

```
df_temp['feature_formula'] = (
    5.9 * df_temp['study_hours'] +
    0.3 * df_temp['class_attendance'] +
    1.4 * df_temp['sleep_hours'] + 4.7819
)
```

1 Funkcja Celu

W kolejnym kroku skupiono się na trenowaniu modelu przy użyciu regresora XGBoost (gradient boosting drzew decyzyjnych).

```
# Model Training
xgb_params = {
    'n_estimators': 10000,
    'learning_rate': 0.007,
    'max_depth': 7,
    'subsample': 0.8,
    'num_parallel_tree': 2,
    'reg_lambda': 3,
    'colsample_bytree': 0.7,
    'tree_method': 'hist',
    'random_state': 42,
    'early_stopping_rounds': 100,
    'eval_metric': 'rmse',
    'enable_categorical': True,
    'n_jobs': -1
}
```

2 Parametry wejściowe dla regresora XGBoost

Postawiono na długi horyzont boostingowy ($n_estimators = 10000$) z małym krokiem ($learning_rate = 0.007$) oraz umiarkowaną głębokością drzew decyzyjnych ($max_depth = 7$)

Trening wykonano w 5 iteracjach, w których trenowano model na 4/5 danych wejściowych danego foldu, a 1/5 służyła jako podzbiór walidacyjny. Dla każdej iteracji zmierzono wskaźniki RMSE oraz R^2 (dla foldów), a na koniec ustalono wartości tych wskaźników dla całego zbioru danych.

```

test_predictions = []
oof_predictions = np.zeros(len(X))
kf = KFold(n_splits=5, shuffle=True, random_state=42)

importances_list = []

for fold, (train_index, val_index) in enumerate(kf.split(X, y)):
    print(f"\n--- Fold {fold+1} ---")

    X_train_fold, X_val = X.iloc[train_index], X.iloc[val_index]
    y_train_fold, y_val = y.iloc[train_index], y.iloc[val_index]

    model = xgb.XGBRegressor(**xgb_params)

    model.fit(
        X_train_fold,
        y_train_fold,
        eval_set=[(X_val, y_val)],
        verbose=500
    )

    fold_importance = pd.DataFrame()
    fold_importance["feature"] = X.columns
    fold_importance["importance"] = model.feature_importances_
    fold_importance["fold"] = fold + 1
    importances_list.append(fold_importance)

    val_preds = model.predict(X_val)
    oof_predictions[val_index] = val_preds

    rmse = np.sqrt(mean_squared_error(y_val, val_preds))
    r2 = r2_score(y_val, val_preds)

    print(f"Fold {fold+1} RMSE: {rmse:.4f}")
    print(f"Fold {fold+1} R2: {r2:.4f}")

test_preds = model.predict(X_test)
test_predictions.append(test_preds)

```

8. Rezultaty

$R^2 = 0.7235$ oznacza, że model wyjaśnia około 72% wariacji wyników egzaminu, co wskazuje na solidną moc predykcyjną

RMSE = 9.9113 należy interpretować względem skali *exam_score*. Jeśli skala wynosi 0–100, średni błąd rzędu ~10 punktów jest umiarkowany: wystarczy do analiz zbiorowych, ale może być graniczny przy decyzjach indywidualnych

OVERALL RMSE: 9.9113

OVERALL R2: 0.7235

Dobrze spisała się również wyprowadzona ręcznie cecha, stanowiąca funkcję celu, wyprzedzając bazowe cechy zbioru w kontekście użyteczności przy trenowaniu modelu

