Project Akhir Praktikum Big Data

Pengembangan Sistem Prediksi Performa Akademik Mahasiswa

Disusun oleh :

Ahda Rindang Al-Amin

(2311531003)

Dosen Pengampu : Luthfil Khairi, S.Kom, M. Cs.
Mata Kuliah : Big Data

PROGRAM STUDI S1-INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
UNIVERSITAS ANDALAS

TAHUN 2025

Pengembangan Sistem Prediksi Performa Akademik Mahasiswa

1. PENDAHULUAN

1.1 Latar Belakang

Dalam dunia pendidikan modern, kemampuan untuk mengidentifikasi mahasiswa yang berisiko mengalami kegagalan akademik sejak dini adalah hal yang krusial. Faktor-faktor seperti kebiasaan belajar, kehadiran di kelas, hingga kualitas tidur memiliki dampak signifikan terhadap hasil ujian akhir. Namun, menganalisis faktor-faktor ini secara manual seringkali tidak efisien dan bias.

Oleh karena itu, penerapan teknologi *Big Data* dan *Machine Learning* diperlukan untuk membangun model prediktif yang objektif. Proyek ini berfokus pada analisis data variabel demografis dan perilaku belajar mahasiswa untuk memprediksi skor ujian (*Exam Score*).

1.2 Tujuan Proyek

a) Melakukan eksplorasi data (*Exploratory Data Analysis*) untuk memahami korelasi antara kebiasaan mahasiswa dengan nilai ujian.
b) Membangun dan membandingkan kinerja beberapa model *Machine Learning* dengan metode regresi dalam memprediksi skor numerik.
c) Mengoptimalkan performa model melalui *Hyperparameter Tuning*.
d) Men-*deploy* model terbaik ke dalam aplikasi web interaktif menggunakan Streamlit.

2. *DATA ACQUISITION & PREPROCESSING*

2.1 Deskripsi *Dataset*

Dataset yang digunakan dalam proyek ini adalah *dataset* publik yang didapatkan dari website Kaggle dengan judul 'Exam Score Prediction Dataset'. *Dataset* ini memenuhi syarat minimum proyek dengan rincian sebagai berikut:

- **Sumber Data**: Kaggle (https://www.kaggle.com/datasets/kundanbedmutha/exam-score-prediction-dataset)
- **Volume Data**: 20.000 baris.
- **Jumlah Fitur**: 12 kolom, terdiri dari fitur numerik dan kategorikal.
- **Target Variabel**: exam_score

Tabel Rincian Fitur:

Nama Fitur	Tipe Data	Deskripsi
student_id	Numerik	UID tiap mahasiswa
age	Numerik	Usia mahasiswa
gender	Kategorikal	Gender mahasiswa
course	Kategorikal	Program studi
study_hours	Numerik	Durasi belajar harian
class_attendance	Numerik (Persentase)	Persentase kehadiran
internet_access	Kategorikal	Ketersediaan internet
sleep_hours	Numerik	Durasi tidur harian
sleep_quality	Ordinal	Kualitas tidur (*poor/average/good*)
study_method	Kategorikal	Metode belajar utama
facility_rating	Ordinal	Kualitas fasilitas kampus (*low/medium/high*)
exam_difficulty	Ordinal	Tingkat kesulitan ujian (*easy/moderate/hard*)
exam_score	Numerik (Target)	Nilai ujian akhir mahasiswa

2.2 Hasil *Exploratory Data Analysis (EDA)*

Berdasarkan *exploratory data analysis* yang dilakukan pada *dataset*, berikut adalah temuan utama:

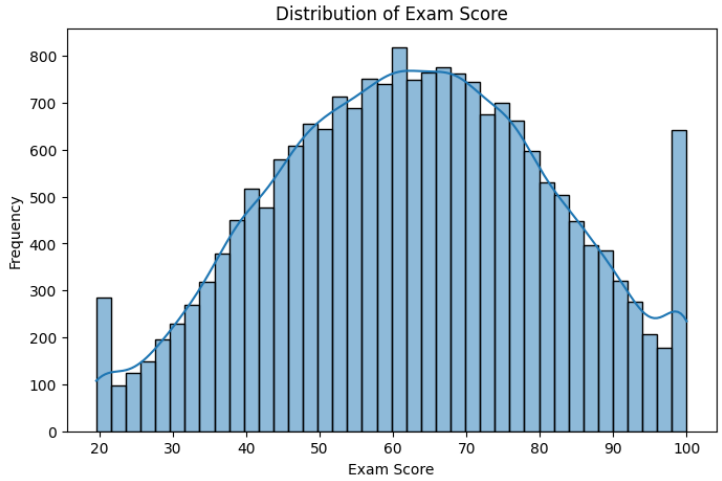
a) Missing Value

Dataset ini tidak memiliki *missing value* sehingga data bisa dilanjutkan ke langkah selanjutnya tanpa adanya kemungkinan bias akibat data yang hilang.

```
Sum of missing values in each column:
student_id          0
age                 0
gender              0
course              0
study_hours         0
class_attendance    0
internet_access     0
sleep_hours         0
sleep_quality       0
study_method        0
facility_rating     0
exam_difficulty     0
exam_score          0
dtype: int64
```

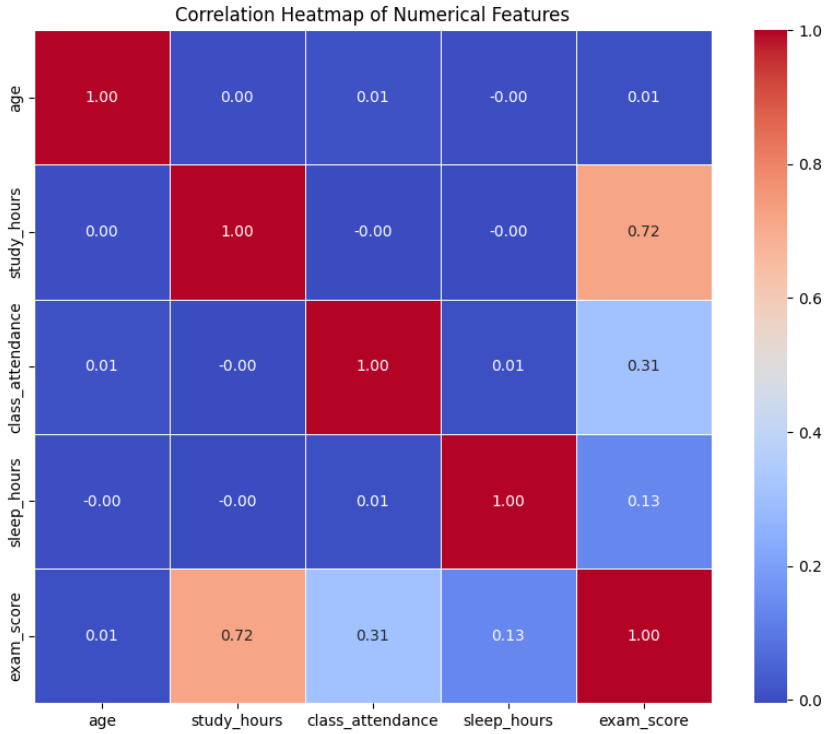
b) Distribusi Target (exam_score)

Data nilai ujian cenderung berdistribusi normal, yang mengindikasikan bahwa model regresi standar cocok digunakan tanpa perlu transformasi logaritma yang ekstrem.



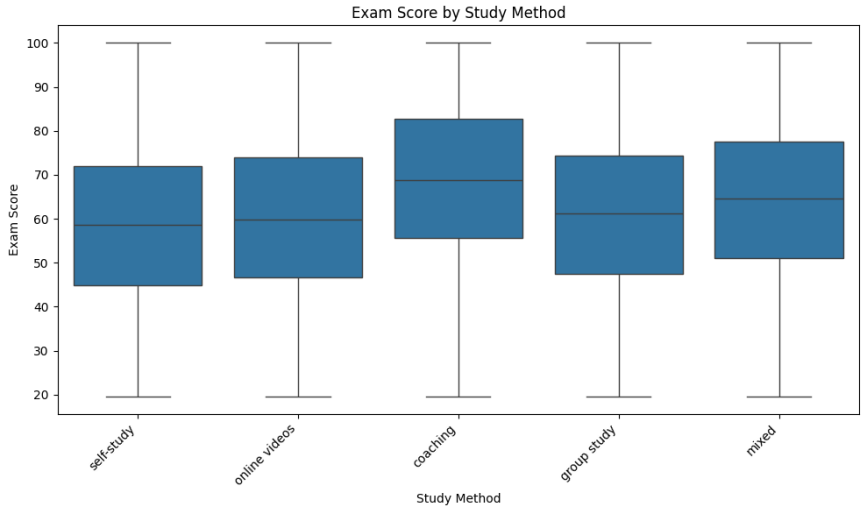
c) Korelasi Antar Fitur

Menggunakan *Heatmap Correlation*, ditemukan bahwa fitur study_hours dan class_attendance memiliki korelasi positif yang paling kuat terhadap exam_score. Artinya, semakin tinggi jam belajar dan kehadiran, semakin tinggi nilai ujian.



d) Kelebihan metode belajar *Coaching*

Visualisasi *Box Plot* menunjukkan perbedaan sebaran nilai berdasarkan study_method terhadap exam_score. Mahasiswa dengan metode belajar *Coaching* rata-rata memiliki nilai ujian yang lebih tinggi dibanding metode belajar lainnya.



2.3 Langkah *Data Preprocessing*

Sebelum pemodelan, data mentah diproses melalui tahapan berikut:

a) *Data Cleaning*

Menghapus kolom student_id karena merupakan *unique identifier* yang tidak memiliki pola prediktif.

```
# Drop ID
df_processed = df.drop('student_id', axis=1)
```

b) *Feature Engineering & Transformation (Pipeline)*

Menggunakan ColumnTransformer dari Scikit-Learn untuk menangani tipe data yang berbeda secara otomatis:

- **Ordinal Encoding**: Diterapkan pada sleep_quality, facility_rating, dan exam_difficulty karena data ini memiliki urutan tingkatan.
- **One-Hot Encoding**: Diterapkan pada gender, course, internet_access, dan study_method karena data ini bersifat nominal (tidak berurutan).
- **Standard Scaling**: Diterapkan pada fitur numerik (study_hours, class_attendance, age) agar semua fitur memiliki skala yang sama (mean=0, variance=1), yang penting untuk performa Linear Regression.

```
# Define columns
ordinal_cols = ['sleep_quality', 'facility_rating', 'exam_difficulty']
onehot_cols = ['gender', 'course', 'internet_access', 'study_method']
numeric_cols_to_scale = ['study_hours', 'class_attendance', 'age']

# Define categories
sleep_quality_categories = ['poor', 'average', 'good']
```

```
facility_rating_categories = ['low', 'medium', 'high']
exam_difficulty_categories = ['easy', 'moderate', 'hard']

categories_list = [
    sleep_quality_categories,
    facility_rating_categories,
    exam_difficulty_categories
]

# Create Preprocessor
preprocessor = ColumnTransformer(
    transformers=[
        ('ord', OrdinalEncoder(categories=categories_list), ordinal_cols),
        ('onehot', OneHotEncoder(handle_unknown='ignore'), onehot_cols),
        ('scaler', StandardScaler(), numeric_cols_to_scale)
    ],
    remainder='passthrough'
)
```

c) *Data Splitting*

Data dibagi menjadi Data Latih (*Train*) dan Data Uji (*Test*) dengan proporsi 80:20 menggunakan train_test_split

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

3. PEMODELAN DAN EVALUASI

3.1 Algoritma yang Digunakan

Tiga algoritma regresi dipilih untuk eksperimen ini:

a) Linear Regression

Model Linear Regression digunakan sebagai *Baseline Model*. Model ini sederhana, cepat, dan mudah diinterpretasikan untuk melihat hubungan linear antar variabel.

b) Random Forest Regressor

Model Random Forest Regressor merupakan model *ensemble* berbasis *bagging* yang mampu menangkap hubungan non-linear yang kompleks dan lebih tahan terhadap outliers.

c) XGBoost Regressor:

XGBoost menggunakan algoritma berbasis *gradient boosting* yang dikenal memiliki performa sangat tinggi (*State-of-the-Art*) pada data tabular terstruktur.

```
# 1. Linear Regression Pipeline
pipeline_lr = Pipeline([
    ('preprocessor', preprocessor),
    ('regressor', LinearRegression())
])

# 2. Random Forest Pipeline
pipeline_rf = Pipeline([
    ('preprocessor', preprocessor),
    ('regressor', RandomForestRegressor(random_state=42, n_jobs=-1))
])

# 3. XGBoost Pipeline
pipeline_xgb = Pipeline([
    ('preprocessor', preprocessor),
    ('regressor', XGBRegressor(random_state=42, n_jobs=-1, tree_method='hist'))
])

models = {
    'Linear Regression': pipeline_lr,
    'Random Forest': pipeline_rf,
    'XGBoost': pipeline_xgb
}
```

3.2 *Hyperparameter Tuning*

Optimasi model dilakukan menggunakan GridSearchCV dengan 3-fold Cross Validation pada data latih.

a) Linear Regression

Tidak dilakukan *hyperparameter tuning*, karena model ini hanya menghitung koefisien dengan metode analitik, tidak ada parameter pada model ini.

b) Random Forest

Parameter yang diuji meliputi n_estimators (100, 200), max_depth (10, 20), dan min_samples_split.

```
# --- Random Forest Tuning ---
param_grid_rf = {
    'regressor__n_estimators': [100, 200],
    'regressor__max_depth': [10, 20],
    'regressor__min_samples_split': [2, 5]
}

grid_search_rf = GridSearchCV(
```

```
    pipeline_rf,
    param_grid_rf,
    cv=3,
    scoring='r2',
    n_jobs=-1,
    verbose=1
)
grid_search_rf.fit(X_train, y_train)
print(f"Best RF R2: {grid_search_rf.best_score_:.4f}")
```

Output : **Best RF R2: 0.7066**

c) XGBoost

Parameter yang diuji meliputi learning_rate (0.05, 0.1), n_estimators, dan max_depth.

```
# --- XGBoost Tuning ---
param_grid_xgb = {
    'regressor__n_estimators': [100, 200],
    'regressor__learning_rate': [0.05, 0.1],
    'regressor__max_depth': [5, 10]
}

grid_search_xgb = GridSearchCV(
    pipeline_xgb,
    param_grid_xgb,
    cv=3,
    scoring='r2',
    n_jobs=-1,
    verbose=1
)
grid_search_xgb.fit(X_train, y_train)
print(f"Best XGB R2: {grid_search_xgb.best_score_:.4f}")
```

Output : **Best XGB R2: 0.7209**

3.3 Tabel Perbandingan Evaluasi

Evaluasi dilakukan pada data uji menggunakan metrik MAE (*Mean Absolute Error*), RMSE (*Root Mean Squared Error*), dan R^2 *Score*.

Model	**MAE**	**RMSE**	**R^2 Score**
Linear Regression	7.862055	9.770710	0.733107
Random Forest (default)	8.315913	10.312769	0.702672
XGBoost (default)	8.328018	10.422202	0.696329
Random Forest (tuned)	(tidak dihitung)	(tidak dihitung)	0.7066
XGBoost (tuned)	(tidak dihitung)	(tidak dihitung)	0.7209

3.4 Analisis Model Terbaik

Berdasarkan tabel evaluasi diatas, model Linear Regression dipilih sebagai model terbaik karena memiliki R^2 *Score* tertinggi serta MAE dan RMSE yang terendah. Artinya model Linear Regression mampu menjelaskan sekitar 73,3% varians dari data nilai ujian.

4. *DEPLOYMENT* APLIKASI STREAMLIT

5.1 Konsep *Pipeline Deployment*

Deployment dilakukan menggunakan *framework* Streamlit (file app.py). Aplikasi ini memuat model yang telah dilatih (.joblib) yang di dalamnya sudah terbungkus *Preprocessing Pipeline*. Ini sangat penting karena data input dari pengguna (yang mentah) akan otomatis melalui proses *scaling* dan *encoding* yang sama persis dengan data latih, mencegah *data leakage* atau error dimensi.

4.2 Antarmuka Pengguna

Desain aplikasi dibuat *user-friendly* dengan fitur:

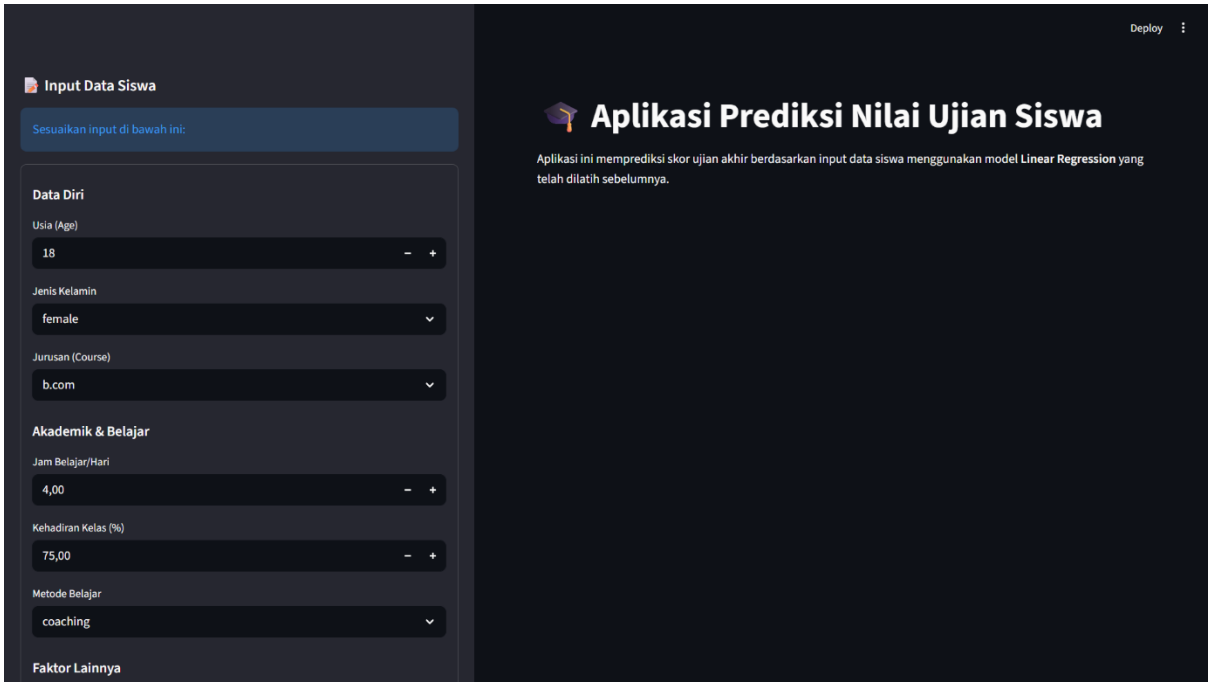
a) ***Sidebar Input***: Pengguna memasukkan data variabel (seperti *slider* untuk Jam Belajar, *dropdown* untuk Jurusan) di sidebar pada sisi kiri halaman.

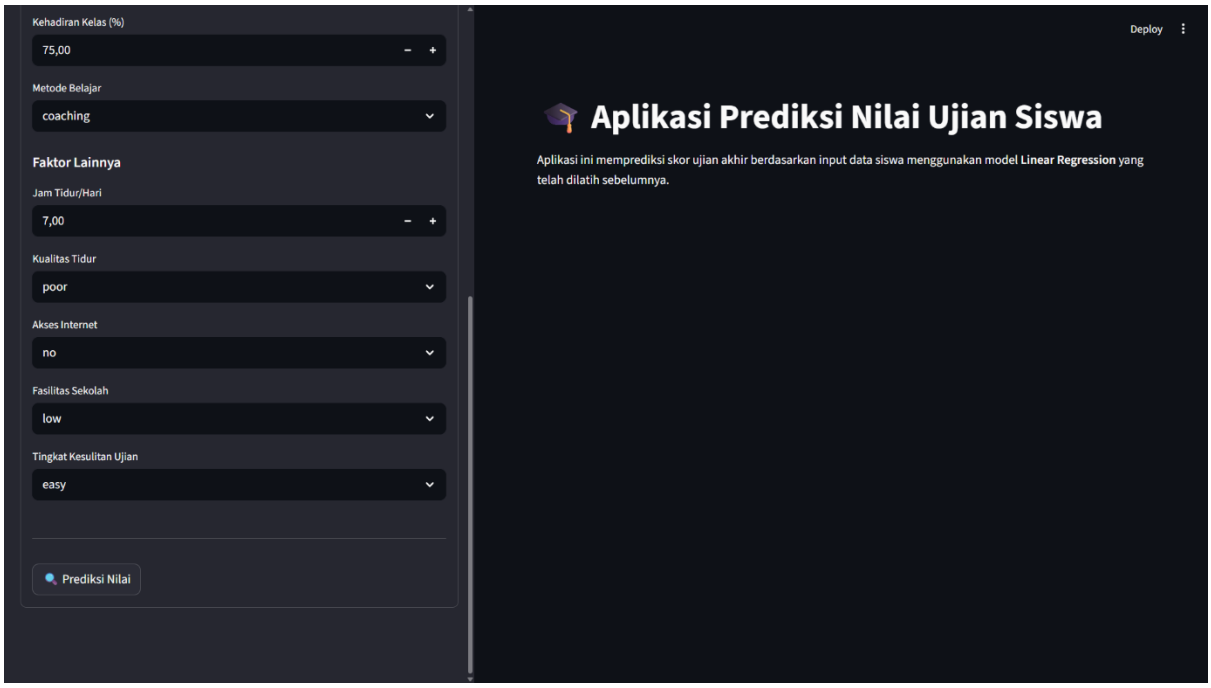
b) ***Main Display***: Aplikasi menampilkan judul dan deskripsi dari proyek, tabel konfirmasi data input, dan hasil prediksi yang menonjol.

c) ***Conditional Formatting***: Aplikasi memberikan umpan balik visual (Pesan "Luar Biasa!" berwarna hijau jika nilai > 80, atau "Perlu Perhatian" jika nilai < 60).

4.3 *Screenshot* Aplikasi

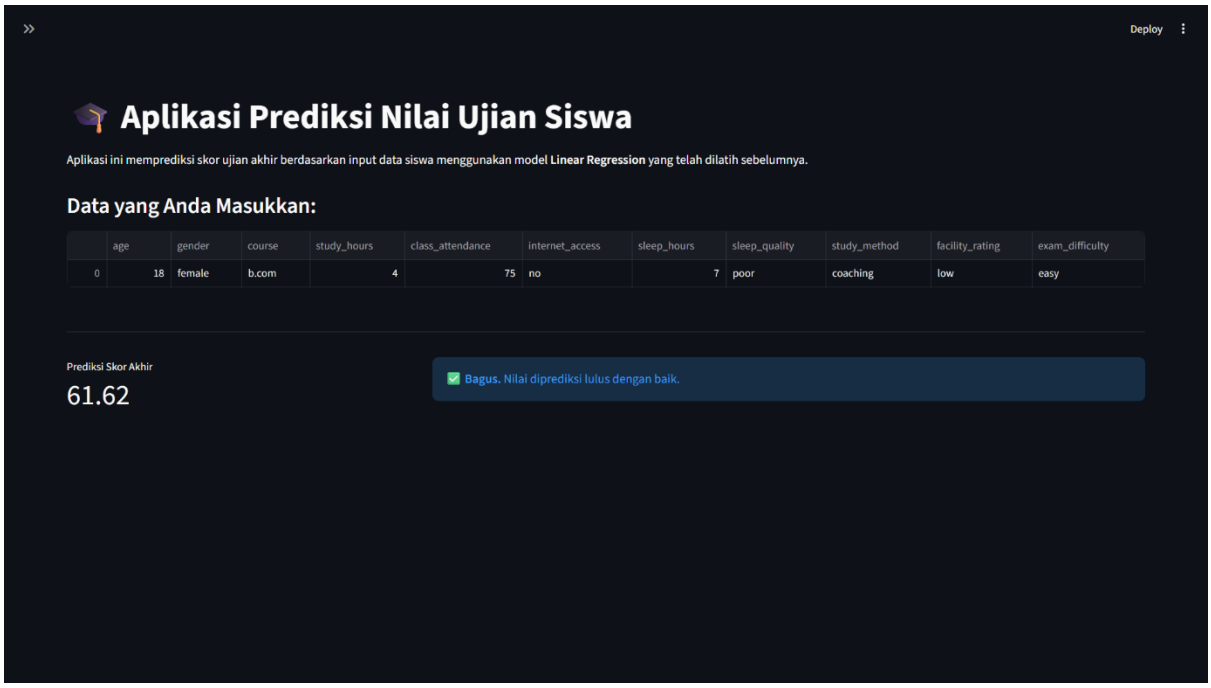
a) Tampilan *Input* Data (*Sidebar & Form*)





Keterangan: Pengguna dapat menggeser slider untuk mengisi data-data fitur sebagai input model prediksi.

b) Hasil Prediksi



Keterangan: Model menampilkan kembali data yang diinput dalam bentuk tabel, dan dibawahnya ditampilkan angka prediksi nilai ujian beserta kalimat umpan balik sesuai prediksi nilai yang didapatkan.

4.4 Instruksi Penggunaan

a) Buka https://prediksi-nilai-ujian.streamlit.app pada *browser*. Jika tidak bisa dibuka atau terjadi *error*, unduh file-file yang dibutuhkan yaitu file app.py, Exam_Score_Prediction.csv, dan file Linear_Regression_(Default)_model.joblib yang bisa diakses di *repository* GitHub pada *link* berikut: https://github.com/ahdarin/prediksi-nilai-ujian.

b) Pastikan file app.py, Exam_Score_Prediction.csv, dan file model .joblib berada dalam satu folder.

c) Buka terminal dan jalankan perintah: streamlit run app.py.

d) Aplikasi akan terbuka di *browser* (localhost).

e) Isi parameter mahasiswa pada menu *sidebar* sebelah kiri, lalu sistem akan otomatis menghitung prediksi nilai.

5. KESIMPULAN DAN SARAN

5.1 Kesimpulan

Proyek ini berhasil mengimplementasikan siklus hidup Big Data secara *end-to-end*. Dimulai dari pemahaman data melalui EDA, ditemukan bahwa data jam belajar dan kehadiran adalah kunci utama keberhasilan akademik. Proses pemodelan menunjukkan bahwa algoritma Linear Regression memberikan akurasi yang paling optimal dengan R^2 *score* 73,3%. Implementasi Streamlit berhasil menjembatani kompleksitas model matematika menjadi aplikasi yang mudah digunakan oleh siapa saja.

5.2 Saran

a) **Pengayaan Data**

Perlu adanya penambahan fitur eksternal seperti latar belakang pendidikan orang tua atau jarak rumah ke sekolah mungkin dapat meningkatkan akurasi.

b) ***Model Complexity***

Mencoba algoritma lain yang lebih kompleks seperti *Deep Learning* (*Neural Networks*) yang lebih bagus jika jumlah data bertambah signifikan (>100.000 baris).