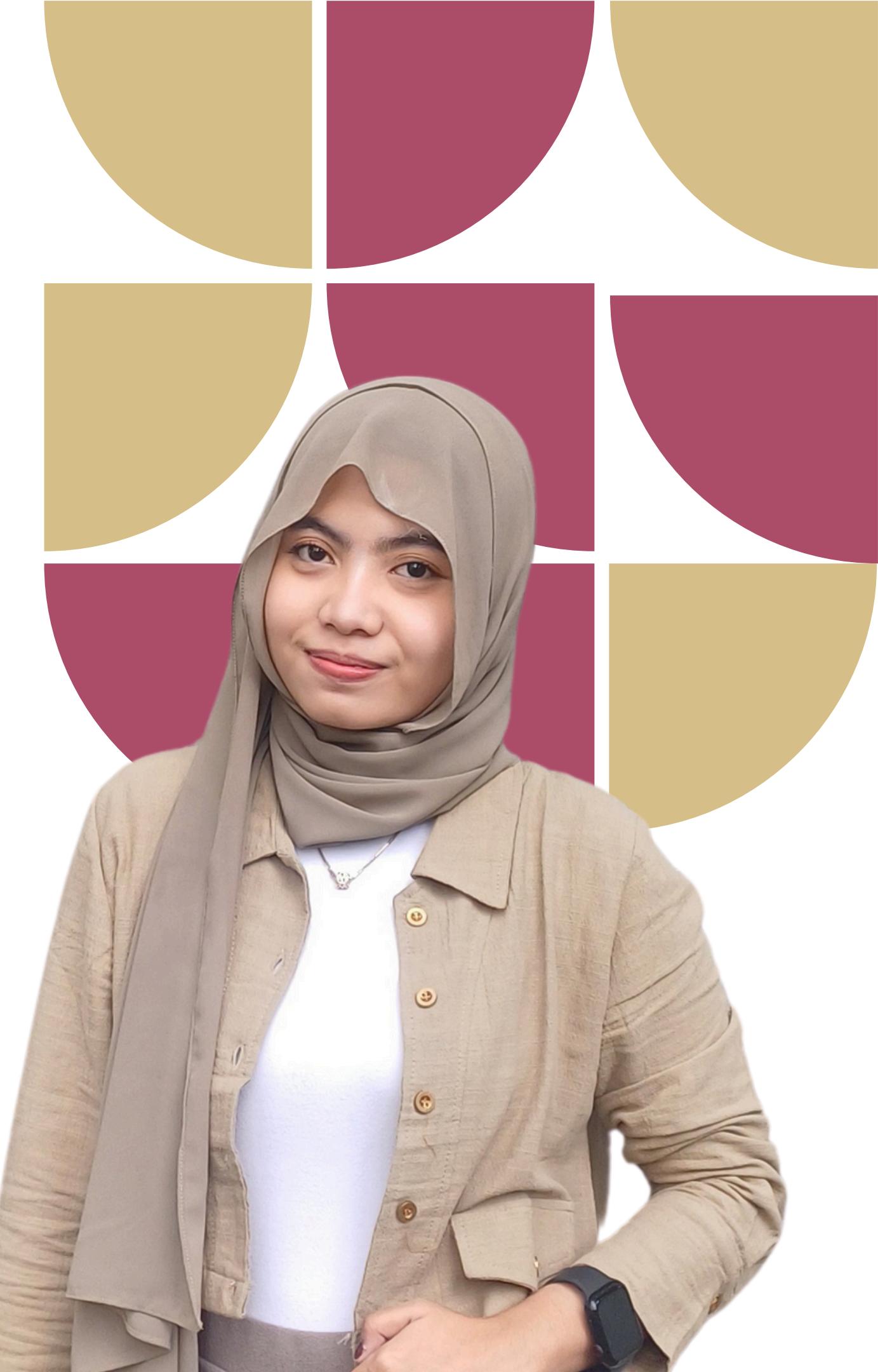


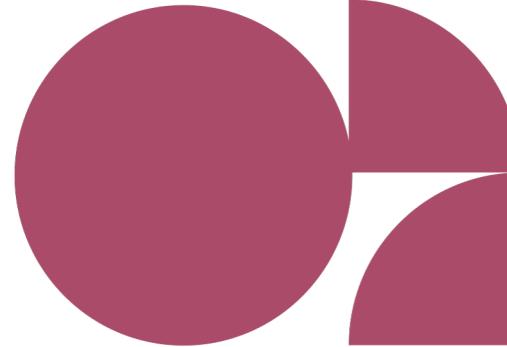


Digital Skill Fair 40.0 - Data Science| Dibimbang

# Data-Driven Insights on Student Admissions: Patterns Toward Equitable Education

→ Anindya Meyla K.S.





# Overview

01. Data Understanding

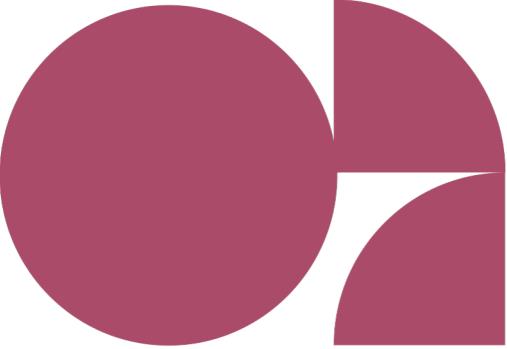
02. Data Preparation

03. Exploratory Data Analysis (EDA)

04. Feature Engineering

05. Classification Model





# Data Understanding

Data yang digunakan dalam analisis klasifikasi ini diperoleh dari Kaggle, yaitu dataset Student Admission Records. Dataset ini mencakup data aplikasi mahasiswa untuk program pascasarjana, dengan total 157 baris dan 7 variabel utama, dengan keterangan seperti pada Tabel berikut.

Variabel	Keterangan	Keterangan
Name	Nama lengkap mahasiswa	Kategorikal
Age	Usia mahasiswa	Numerik
Gender	Jenis kelamin mahasiswa	Kategorikal
Admission Test Score	Skor ujian masuk (0-100)	Numerik
High School Percentage	Persentase nilai SMA	Numerik
City	Kota tempat tinggal mahasiswa	Kategorikal
Admission Status	Status diterima atau tidak	Kategorikal

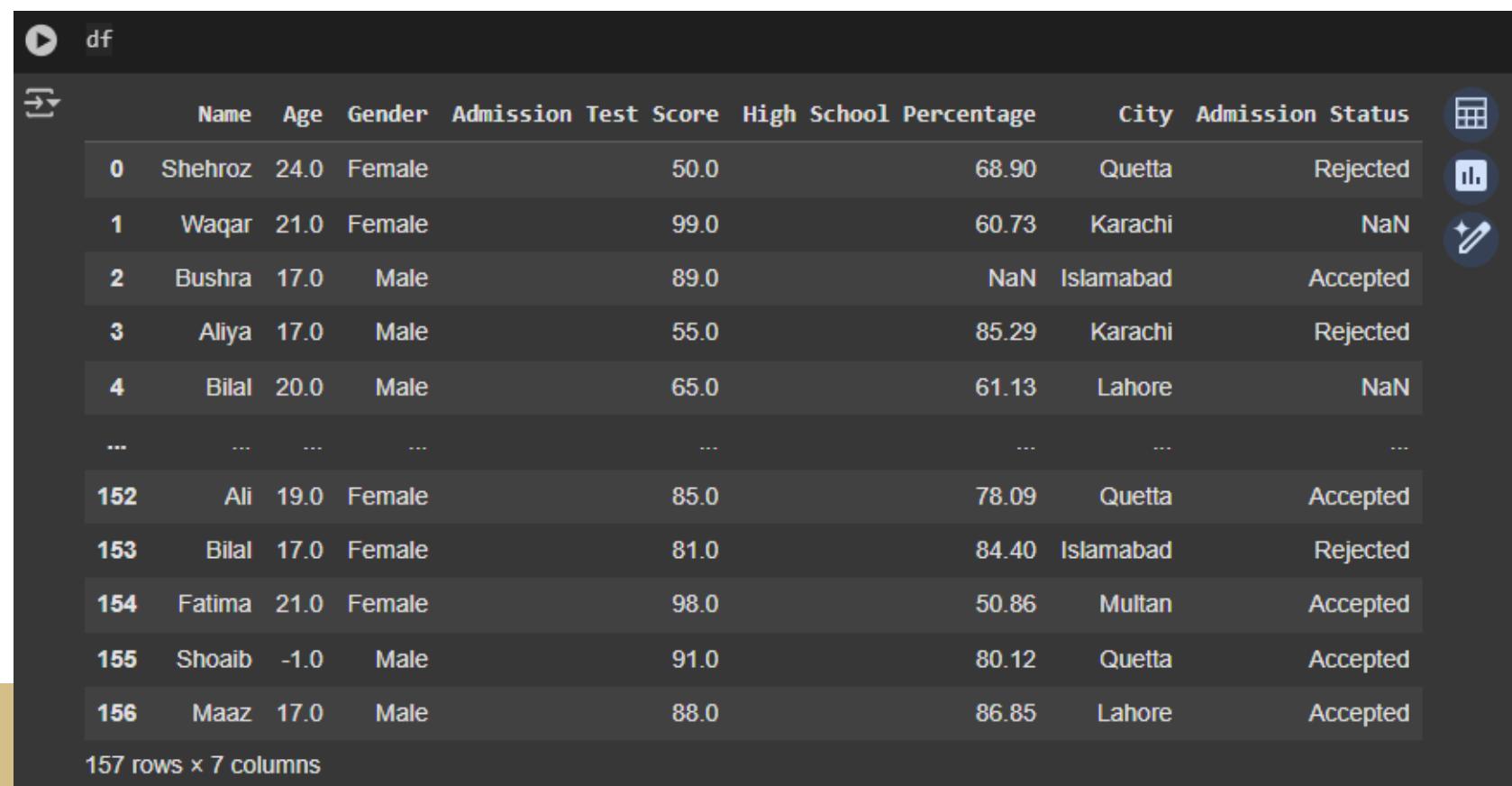


# Data Preparation

## 1. Membaca Dataset

```
[172] # Mengimport library pandas untuk membaca dataset
      import pandas as pd

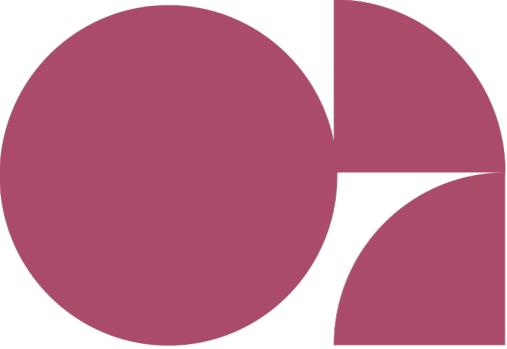
      # baca dataset csv menggunakan syntax pd.read_csv()
      df = pd.read_csv('/content/student_admission_record_dirty.csv')
```



	Name	Age	Gender	Admission Test Score	High School Percentage	City	Admission Status
0	Shehroz	24.0	Female	50.0	68.90	Quetta	Rejected
1	Waqar	21.0	Female	99.0	60.73	Karachi	NaN
2	Bushra	17.0	Male	89.0	NaN	Islamabad	Accepted
3	Aliya	17.0	Male	55.0	85.29	Karachi	Rejected
4	Bilal	20.0	Male	65.0	61.13	Lahore	NaN
...	...	...	...	...	...	...	...
152	Ali	19.0	Female	85.0	78.09	Quetta	Accepted
153	Bilal	17.0	Female	81.0	84.40	Islamabad	Rejected
154	Fatima	21.0	Female	98.0	50.86	Multan	Accepted
155	Shoaib	-1.0	Male	91.0	80.12	Quetta	Accepted
156	Maaz	17.0	Male	88.0	86.85	Lahore	Accepted

157 rows × 7 columns

**Membaca dataset** dari file CSV agar dapat diolah dengan Python. Proses ini menggunakan pustaka pandas, yang mempermudah pemrosesan data dalam bentuk tabel (dataframe).

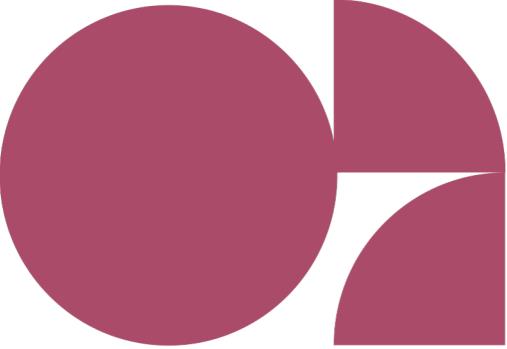


# Data Preparation

```
▼ 2. Ringkasan Data  
[174] df.info()  
  
→ <class 'pandas.core.frame.DataFrame'>  
RangeIndex: 157 entries, 0 to 156  
Data columns (total 7 columns):  
 #   Column      Non-Null Count  Dtype     
 ---    
 0   Name        147 non-null    object    
 1   Age         147 non-null    float64  
 2   Gender      147 non-null    object    
 3   Admission Test Score 146 non-null    float64  
 4   High School Percentage 146 non-null    float64  
 5   City         147 non-null    object    
 6   Admission Status     147 non-null    object    
dtypes: float64(3), object(4)  
memory usage: 8.7+ KB
```

**Ringkasan Data**, Pada tahap ini dilakukan eksplorasi awal untuk memahami struktur dataset. Fungsi info() dari pandas digunakan untuk:

- ✓ menampilkan jumlah baris (entries)
- ✓ melihat nama kolom
- ✓ melihat tipe data setiap kolom
- ✓ memeriksa jumlah data yang non-null



# Data Preparation

## 3. Mengecek Data yang Hilang (Missing Value)

```
# mengecek missing value  
df.isna().sum()
```

```
          0  
Name      10  
Age       10  
Gender    10  
Admission Test Score  11  
High School Percentage 11  
city      10  
Admission Status     10  
  
dtype: int64
```

```
[177] # cek statiscial summary  
df['City'].describe()
```

```
          City  
count    147  
unique     7  
top      Quetta  
freq      30  
  
dtype: object
```

**Mengecek data hilang (missing value), dapat dilihat jumlah nilai kosong di setiap kolom:**

- name hilang 10 data
- age hilang 10 data
- gender hilang 10 data
- admission test score hilang 11 data
- high school percentage hilang 11 data
- city hilang 10 data
- admission status hilang 10 data



# Data Preparation

```
# menggunakan boxplot untuk mendeteksi outlier pada ketiga kolom numerik
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

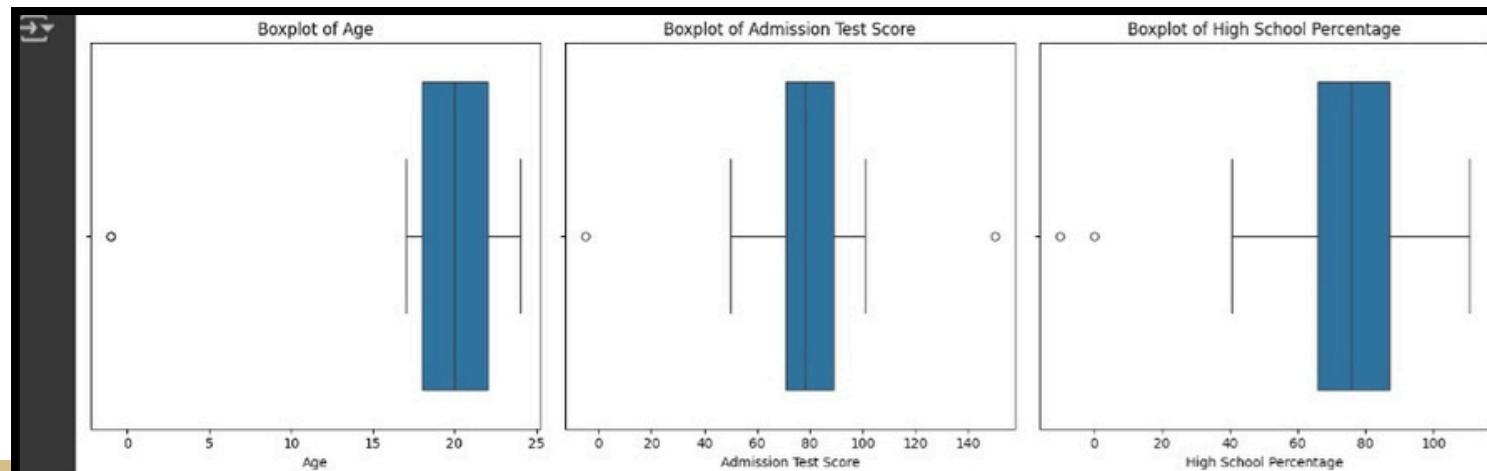
# membuat subplot untuk 3 boxplot dalam 1 figure
plt.figure(figsize=(15, 5))

# boxplot Age
plt.subplot(1, 3, 1)
sns.boxplot(x=df['Age'])
plt.title('Boxplot of Age')

# boxplot Admission Test Score
plt.subplot(1, 3, 2)
sns.boxplot(x=df['Admission Test Score'])
plt.title('Boxplot of Admission Test Score')

# boxplot High School Percentage
plt.subplot(1, 3, 3)
sns.boxplot(x=df['High School Percentage'])
plt.title('Boxplot of High School Percentage')

# mengatur layout agar tidak overlap
plt.tight_layout()
plt.show()
```



**Mengecek outlier**, terlihat bahwa variabel 'Age' 'Admission Test Score' dan 'High School Percentage' memiliki outlier, terutama di bawah nilai 0 dan di atas 100.

Dalam konteks klasifikasi atau pemodelan supervised learning, outlier seperti ini perlu diperiksa lebih lanjut apakah merupakan kesalahan input. Jika ya, maka outlier sebaiknya dibersihkan atau diimputasi agar tidak mempengaruhi hasil model secara ekstrem.



# Data Preparation

```
[178] # Mengatasi missing value
for column in df.columns:
    if df[column].dtype == 'object':
        # Jika kolom bertipe object, isi dengan mode
        df[column].fillna(df[column].mode()[0], inplace=True)
    else:
        # Jika kolom bertipe numerik, isi dengan mean
        df[column].fillna(df[column].mean(), inplace=True)
```

```
# cek kembali missing value
df.isna().sum()
```

	0
Name	0
Age	0
Gender	0
Admission Test Score	0
High School Percentage	0
City	0
Admission Status	0
<b>dtype:</b>	<b>int64</b>

```
[180] df.info()
```

```
→ <class 'pandas.core.frame.DataFrame'>
RangeIndex: 157 entries, 0 to 156
Data columns (total 7 columns):
 #   Column           Non-Null Count  Dtype  
--- 
 0   Name             157 non-null    object  
 1   Age              157 non-null    float64 
 2   Gender            157 non-null    object  
 3   Admission Test Score  157 non-null  float64 
 4   High School Percentage 157 non-null  float64 
 5   City              157 non-null    object  
 6   Admission Status  157 non-null    object  
dtypes: float64(3), object(4)
memory usage: 8.7+ KB
```

**Handling missing value** dengan strategi pengisian berbasis tipe data. Kolom kategorikal diisi dengan modus (nilai yang paling sering muncul), sedangkan kolom numerik diisi dengan mean (rata-rata). Setelah proses imputasi selesai, dipastikan kembali jumlah missing value adalah nol dengan mengecek ulang struktur data.



# Data Preparation

## ▼ 4. Mengatasi Duplikat Data

```
[181] # Mengecek apakah ada duplicate di seluruh kolom  
    check_duplicate = df.duplicated().sum()  
  
    print(f"Jumlah data yang duplikat = {check_duplicate}")
```

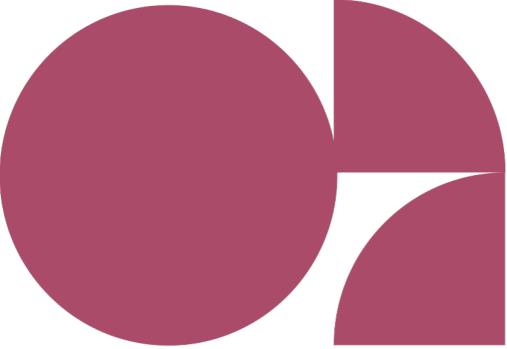
→ Jumlah data yang duplikat = 7

```
[182] # Handling duplicate  
    df = df.drop_duplicates()
```

```
▶ # Mengecek duplicate setelah di-handle  
    handle_duplicate = df.duplicated().sum()  
  
    print(f"Jumlah data yang duplikat = {handle_duplicate}")
```

→ Jumlah data yang duplikat = 0

**Handling data duplikat**, dengan total 7 baris teridentifikasi, kemudian dihapus menggunakan fungsi `drop_duplicates()` sehingga jumlah data duplikat menjadi nol, memastikan tidak ada redundansi di dalam dataset.



# Data Preparation

```
▶ from sklearn.preprocessing import RobustScaler
  # inisiasi RobustScaler
  scaler = RobustScaler()

  # standarisasi untuk ketiga numerical features
  numerical_features = ['Age', 'Admission Test Score', 'High School Percentage']

  # fit dan transform sekaligus
  scaled_data = scaler.fit_transform(df[numerical_features])

  # assignment menggunakan .loc untuk menghindari warning
  df.loc[:, 'Age_scaled'] = scaled_data[:, 0]
  df.loc[:, 'Admission Test Score_scaled'] = scaled_data[:, 1]
  df.loc[:, 'High School Percentage_scaled'] = scaled_data[:, 2]

  # melihat hasil transformasi
  df[['Age', 'Age_scaled', 'Admission Test Score', 'Admission Test Score_scaled',
       'High School Percentage', 'High School Percentage_scaled']].head()
```

	Age	Age_scaled	Admission Test Score	Admission Test Score_scaled	High School Percentage	High School Percentage_scaled
0	24.0	1.00	50.0	-1.555556	68.900000	-0.320677
1	21.0	0.25	99.0	1.166667	60.730000	-0.706829
2	17.0	-0.75	89.0	0.611111	75.684726	0.000000
3	17.0	-0.75	55.0	-1.277778	85.290000	0.453989
4	20.0	0.00	65.0	-0.722222	61.130000	-0.687923

**Melakukan standarisasi**, menggunakan RobustScaler dari sklearn. Metode ini dipilih karena lebih tahan terhadap outliers dibandingkan StandardScaler, dengan menggunakan median dan IQR (Interquartile Range) untuk scaling sehingga outliers tidak mempengaruhi hasil standardisasi secara signifikan.

Hasil standardisasi menghasilkan data dengan median=0 dan sebaran yang proporsional terhadap IQR.



# Feature Engineering

Terdapat tiga feature engineering yang dilakukan, yaitu dengan encoding categorical variables yaitu

## 1. Gender Encoding

Fitur ini mengubah data kategorik Gender menjadi format numerik untuk mempermudah pemrosesan oleh algoritma machine learning. Outputnya berupa data numerik biner dengan dua nilai yaitu:

1 -> Female dan 0 -> Male

## 2. Admission Status Encoding

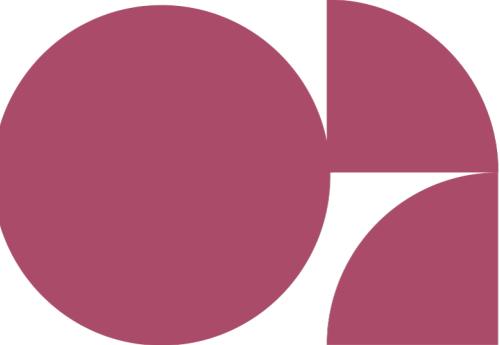
Fitur ini merupakan encoding untuk target variable yang mengubah status penerimaan menjadi format numerik biner. Outputnya berupa data numerik biner dengan dua nilai yaitu:

1 → Accepted (mahasiswa diterima) dan 0 → Rejected (mahasiswa ditolak)

## 3. City Encoding

Fitur ini mengubah data kategorik City menjadi multiple binary variables menggunakan One-Hot Encoding. Setiap kota direpresentasikan sebagai kolom biner, yaitu:

1 -> berasal dari kota tersebut, dan 0 -> berasal dari kota lain.



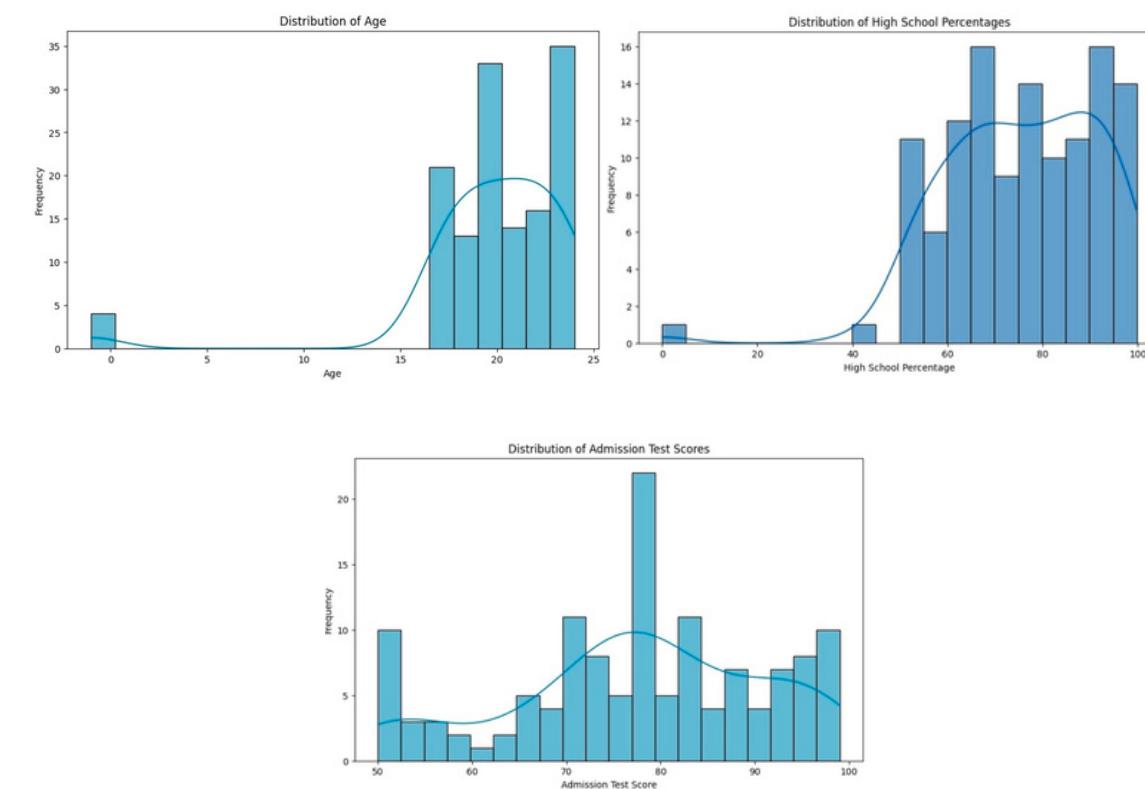
# Exploratory Data Analysis (EDA)

```
[176] # cek statiscal summary
df.describe()
```

	Age	Admission Test Score	High School Percentage
count	147.000000	146.000000	146.000000
mean	19.680272	77.657534	75.684726
std	4.540512	16.855343	17.368014
min	-1.000000	-5.000000	-10.000000
25%	18.000000	68.250000	65.052500
50%	20.000000	79.000000	77.545000
75%	22.000000	89.000000	88.312500
max	24.000000	150.000000	110.500000

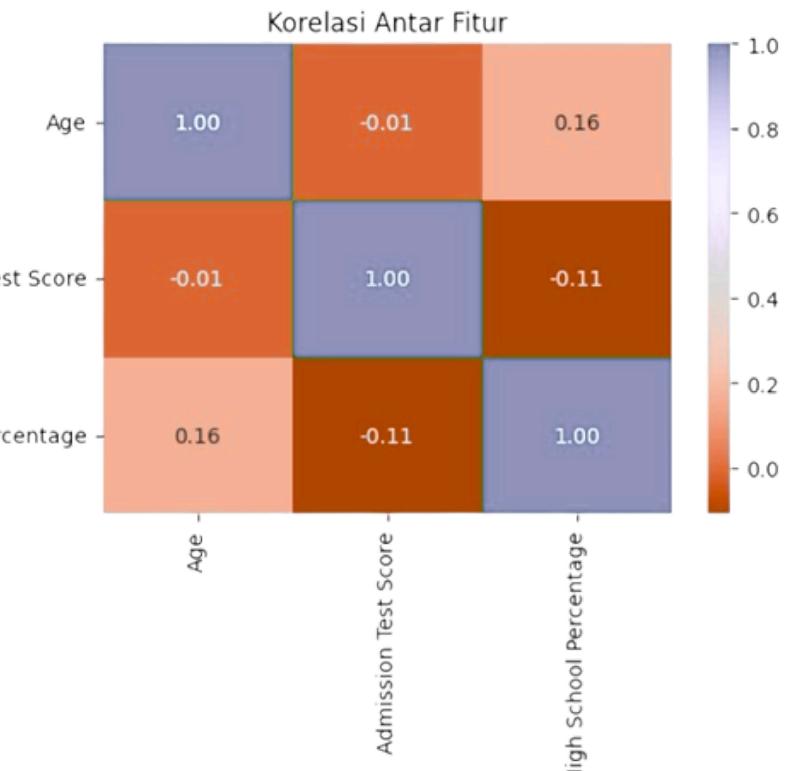
## Statistical Five Summaries

Menampilkan ringkasan dari tiga numerik yang digunakan dalam analisis.



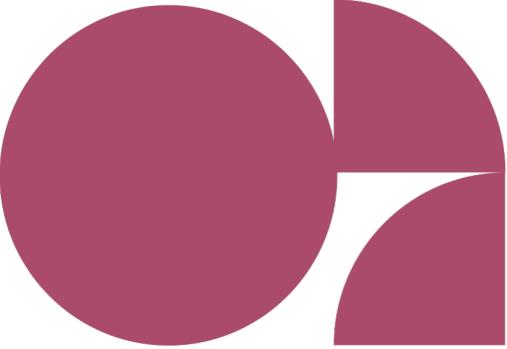
## Distribusi Data

Distribusi *Age* dan Distribusi *High School Percentage* cenderung left skewed, lalu Distribusi *Admission Test Score* cenderung symmetric

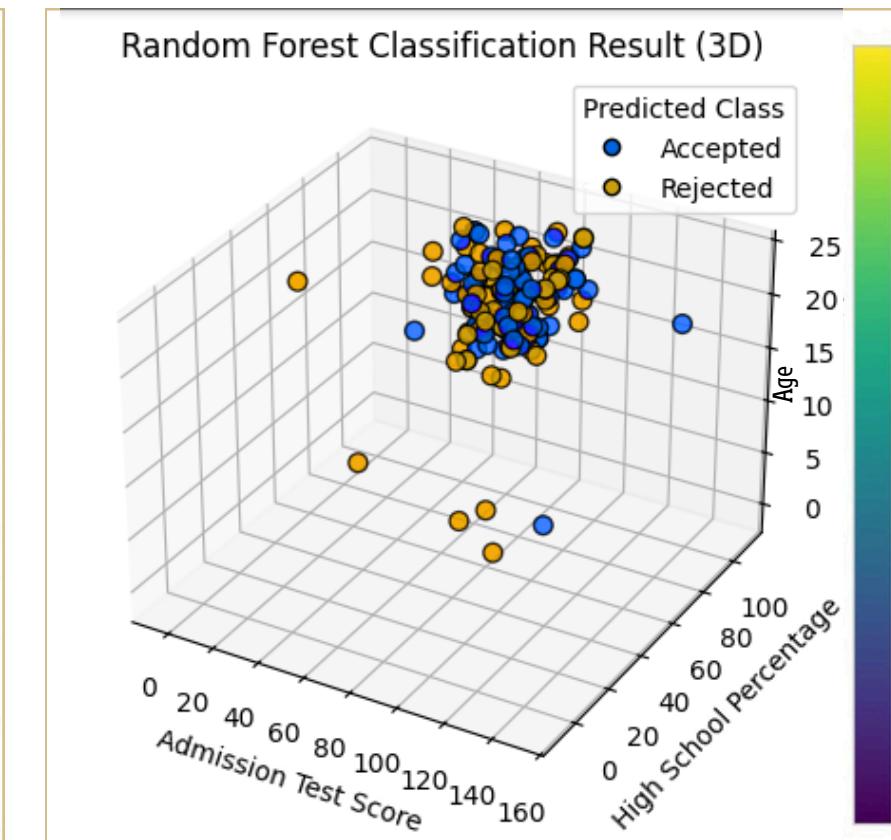
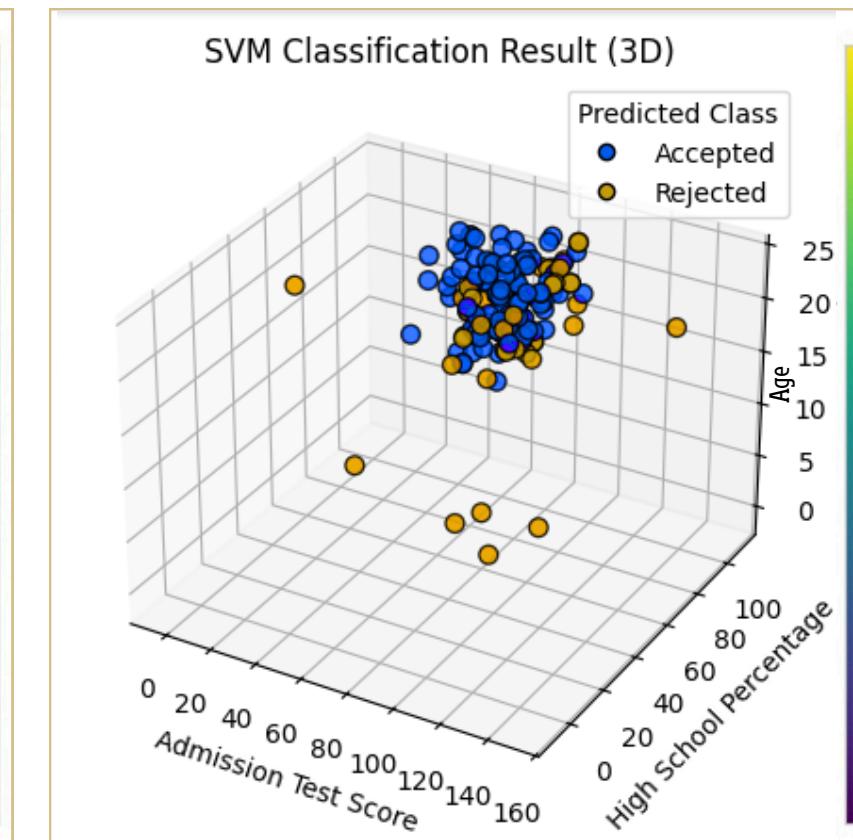
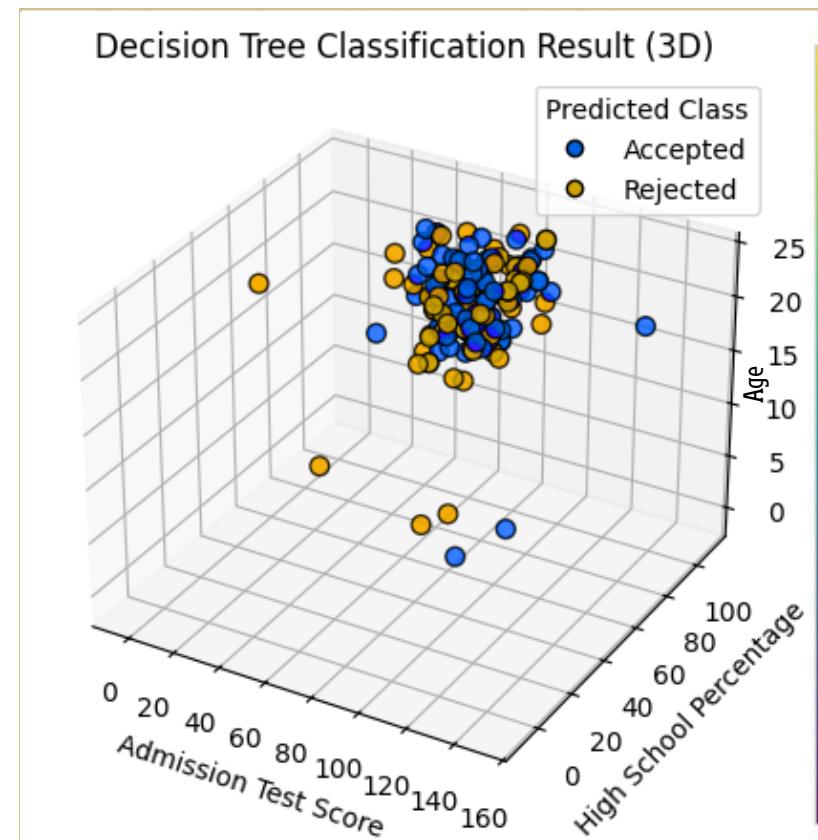
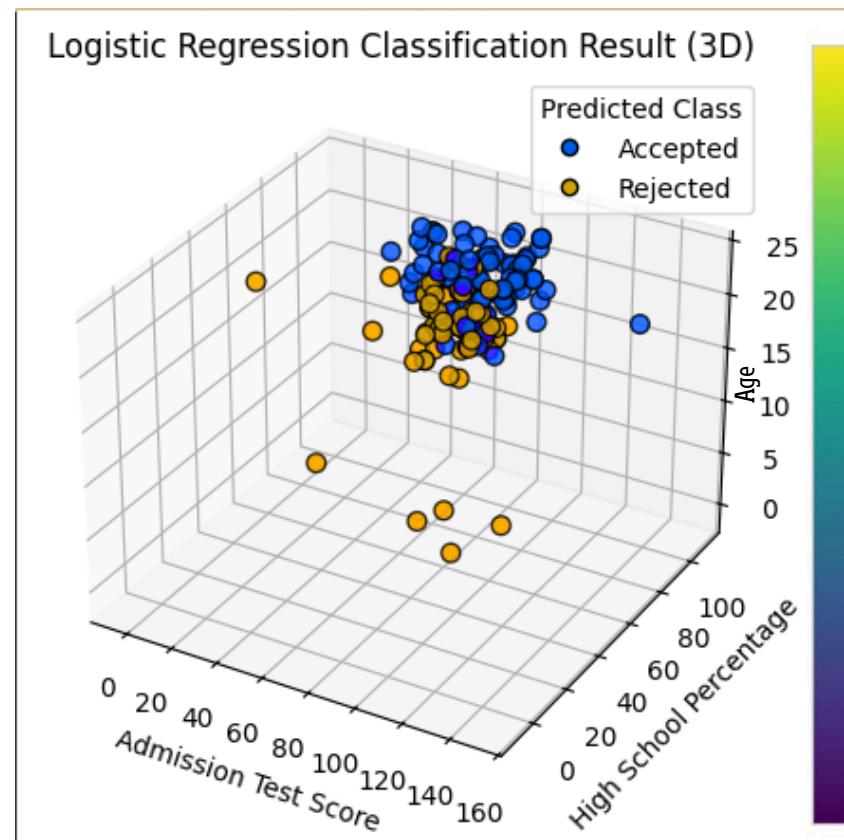


## Korelasi Antar Variabel

Hasil Korelasi rendah (< 0.3), menunjukkan hubungan yang lemah dan tidak signifikan.



# Classification Model

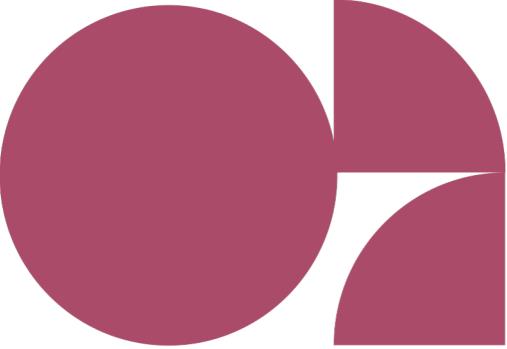


→ Logistic  
Regression

→ Decision Tree

→ SVM

→ Random Forest



# Classification Model

## Model 1: Logistic Regression

Class	Keterangan
Class 0 (Rejected)	Calon mahasiswa dengan nilai tes masuk tinggi, nilai rata-rata SMA tinggi, serta profil demografis (gender, kota) yang lebih sesuai kriteria universitas.
Class 1 (Accepted)	Calon mahasiswa dengan nilai tes masuk tinggi, nilai rata-rata SMA tinggi, serta profil demografis (gender, kota) yang lebih sesuai kriteria universitas.

## Model 2: Decision Tree

Class	Keterangan
Class 0 (Rejected)	Teridentifikasi terutama dari skor tes masuk rendah di bawah ambang batas tertentu.
Class 1 (Accepted)	Memiliki skor tes masuk tinggi dan nilai SMA tinggi, yang melewati batas keputusan algoritma Decision Tree.



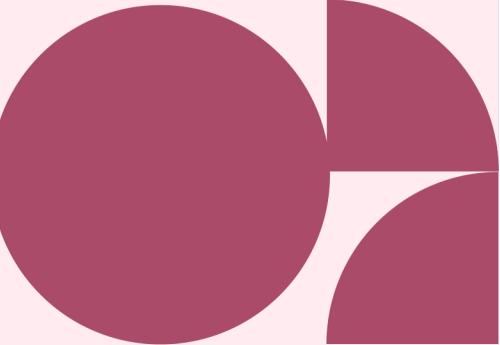
# Classification Model

## Model 3: SVM

Class	Keterangan
Class 0 (Rejected)	Dipisahkan oleh hyperplane sebagai data yang jauh dari margin batas kelas diterima.
Class 1 (Accepted)	Masuk area positif margin hyperplane, menandakan profil akademik diterima.

## Model 4: Random Forest

Class	Keterangan
Class 0 (Rejected)	Konsisten tertolak di sebagian besar pohon keputusan karena kombinasi nilai SMA dan skor tes yang kurang optimal.
Class 1 (Accepted)	Diterima oleh mayoritas voting pohon Random Forest karena skor tes dan nilai SMA tinggi.



# Model Terbaik

Pemilihan model terbaik dilakukan dengan melihat Accuracy, dengan semakin tinggi nilai Accuracy maka semakin baik performa model Classification.

Model Klasifikasi	Akurasi
Logistic Regression	0,47
Decision Tree	0,53
Random Forest	0,67
Support Vector Machine (SVM)	0,50

Berdasarkan evaluasi menggunakan skor accuracy, maka model Random Forest Classification memberikan hasil terbaik untuk dataset ini karena memiliki nilai akurasi tertinggi dibandingkan tiga metode lainnya. Hal ini menunjukkan lebih stabil dan konsisten.



# Terima Kasih

Sampai jumpa di kesempatan berikutnya