In [1]:
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
from sklearn.preprocessing import StandardScaler
from IPython.display import display, HTML
import re
```

In [2]:
```
pip install xlrd==2.0.1
```

```
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
data = pd.read_excel("Dataset M1.xls")
df = data.copy()
```

In [4]:
```
def getting_primary_info(df):
    print("-----------------------------------------------------")
    print("Veri setinin şekli", df.shape)
    print("-----------------------------------------------------")
    print("Veri seti değişken tipleri:\n", df.dtypes)
    print("-----------------------------------------------------")
    print("Veri setinin ilk 5 satırı")
    display(HTML(df.head().to_html()))
    print("-----------------------------------------------------")
    print("Veri setinin istatistiki verileri")
    description = df.describe()
    display(HTML(description.to_html()))
    print("-----------------------------------------------------")
getting_primary_info(df)
```

```
-----------------------------------------------------
Veri setinin şekli (1007, 163)
-----------------------------------------------------
Veri seti değişken tipleri:
 Öğrenci No      object
Cinsiyet       float64
Doğum Yılı     float64
Okul Türü        int64
Sınıf Düzeyi     int64
               ...   
AE               int64
CE-AC          float64
RO-AE            int64
İst              int64
pass             int64
Length: 163, dtype: object
-----------------------------------------------------
Veri setinin ilk 5 satırı
```

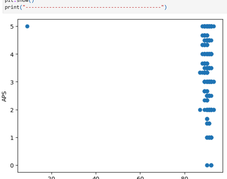
	Öğrenci No	Cinsiyet	Doğum Yılı	Okul Türü	Sınıf Düzeyi	Baba Mesleği	Baba Eğitim Düzeyi	Anne Mesleği	Anne Eğitim Düzeyi	Kardeş Sayısı	Kitap Sayısı	Evde Bilgisayar	Evde İnternet	Evde Çalışma Odası	Sınıf 9 Notu	Sınıf 10 Notu	Sınıf 11 Notu	Not Ortalaması aps	Ders Dinleme Önemi	Ders Çalışma Önemi	Ödev Yapma Önemi	Okul Kursu Önemi	Özel Ders Sıklığı	Dışın Fizik Sıklığı	Dışın OSS Sıklığı	En sevilen konu	Isl
0	1-100	1.0	92.0	3	9	12.0	2.0	12.0	2.0	0.0	2.0	1.0	1.0	1.0	3.0	NaN	NaN	3.0	1.0	1.0	1.0	3.0	1	1.0	1.0	6.0	
1	1-102	2.0	92.0	3	9	12.0	3.0	14.0	2.0	2.0	2.0	1.0	1.0	1.0	2.0	NaN	NaN	2.0	1.0	2.0	1.0	3.0	1	1.0	1.0	6.0	
2	1-103	2.0	92.0	3	9	11.0	5.0	14.0	2.0	2.0	3.0	0.0	0.0	1.0	2.0	NaN	NaN	2.0	1.0	1.0	1.0	1.0	1	1.0	1.0	6.0	
3	1-105	1.0	92.0	3	9	12.0	4.0	12.0	3.0	1.0	3.0	1.0	1.0	1.0	3.0	NaN	NaN	3.0	1.0	1.0	2.0	1.0	1	1.0	1.0	6.0	
4	1-106	2.0	92.0	3	9	15.0	3.0	14.0	2.0	3.0	1.0	1.0	1.0	1.0	1.0	NaN	NaN	1.0	1.0	1.0	1.0	3.0	1	1.0	2.0	NaN	

```
-----------------------------------------------------
Veri setinin istatistiki verileri
```

	Cinsiyet	Doğum Yılı	Okul Türü	Sınıf Düzeyi	Baba Mesleği	Baba Eğitim Düzeyi	Anne Mesleği	Anne Eğitim Düzeyi	Kardeş Sayısı	Kitap Sayısı	Evde Bilgisayar	Evde İnternet	Evde Çalışma Odası	Sınıf 9 Notu	Sınıf 10 Notu	Sınıf 11 Notu	Not Ortalaması aps	D
count	1006.000000	1006.000000	1007.000000	1007.000000	979.000000	993.000000	996.000000	1001.000000	1003.000000	997.000000	1006.000000	1006.000000	1006.000000	974.000000	646.000000	234.000000	978.000000	1004.
mean	1.503976	90.546720	2.488858	9.929494	12.899898	3.853978	13.644578	3.118881	1.569292	3.141424	0.711730	0.457256	0.870775	3.74538	3.431889	4.098291	3.525392	1.
std	0.500233	2.816318	1.28364	0.755361	7.567973	1.210600	4.947786	1.240505	0.929627	0.997517	0.453183	0.498417	0.335615	1.17915	1.315680	1.024882	1.155935	0.
min	1.000000	9.000000	1.000000	9.000000	1.000000	0.000000	1.000000	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	1.
25%	1.000000	90.000000	1.000000	9.000000	10.000000	3.000000	14.000000	2.000000	1.000000	2.000000	0.000000	0.000000	1.000000	3.000000	3.000000	4.000000	3.000000	1.
50%	2.000000	91.000000	3.000000	10.000000	12.000000	4.000000	14.000000	3.000000	1.000000	3.000000	1.000000	0.000000	1.000000	4.000000	4.000000	4.000000	3.666667	1.
75%	2.000000	92.000000	3.000000	11.000000	16.000000	5.000000	14.000000	4.000000	2.000000	4.000000	1.000000	1.000000	1.000000	5.000000	5.000000	5.000000	4.500000	1.
max	2.000000	93.000000	5.000000	11.000000	52.000000	14.000000	45.000000	7.000000	4.000000	4.000000	1.000000	1.000000	1.000000	5.000000	5.000000	5.000000	5.000000	3.

```
-----------------------------------------------------
```

In [5]:
```
plt.scatter(x = "Doğum Yılı" , y = 'Not Ortalaması aps', data = df)
plt.xlabel("yas")
plt.ylabel('APS')
plt.show()
print("-----------------------------------------------------")
```




```
-----------------------------------------------------
```

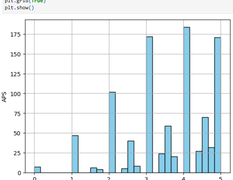
In [6]:
```
df.query(" Doğum Yılı  ==  87")
```

Out[6]:

	Öğrenci No	Cinsiyet	Doğum Yılı	Okul Türü	Sınıf Düzeyi	Baba Mesleği	Baba Eğitim Düzeyi	Anne Mesleği	Anne Eğitim Düzeyi	Kardeş Sayısı	...	c11	c12	CE	RO	AC	AE	CE-AC	RO-AE	İst	pass
532	7-33	2.0	87.0	1	11	15.0	2.0	14.0	0.0	4.0	...	4.0	4.0	26.0	34	25	35	1.0	-1	2	27
824	12-1277	1.0	87.0	1	11	15.0	2.0	14.0	2.0	4.0	...	4.0	4.0	21.0	29	28	42	-7.0	-13	3	13
998	14-x38	2.0	87.0	2	11	12.0	2.0	14.0	2.0	2.0	...	4.0	4.0	38.0	40	40	44	-2.0	-4	3	21

3 rows × 163 columns

In [7]:
```
plt.hist(df["Not Ortalaması aps"], bins=30, color='skyblue', edgecolor='black')
plt.xlabel('Değerler')
plt.ylabel('APS')
plt.grid(True)
plt.show()
```



In [8]:
```
df["Cinsiyet"].value_counts()
df["Cinsiyet"] = df["Cinsiyet"].fillna(1)
```

In [9]:
```
df["En sevilen konu"].value_counts()
df["En sevilen konu"] = df["En sevilen konu"].fillna(12)
```

In [10]:
```
df["Doğum Yılı"].value_counts()
df["Doğum Yılı"] = df["Doğum Yılı"].fillna(90)
```

In [11]:
```
df.drop(columns = ["Sınıf 11 Notu"])
```

Out[11]:

	Öğrenci No	Cinsiyet	Doğum Yılı	Okul Türü	Sınıf Düzeyi	Baba Mesleği	Baba Eğitim Düzeyi	Anne Mesleği	Anne Eğitim Düzeyi	Kardeş Sayısı	...	c11	c12	CE	RO	AC	AE	CE-AC	RO-AE	İst	pass
0	1-100	1.0	92.0	3	9	12.0	2.0	12.0	2.0	0.0	...	4.0	4.0	25.0	26	38	31	-13.0	-5	3	27
1	1-102	2.0	92.0	3	9	12.0	3.0	14.0	2.0	2.0	...	4.0	4.0	69.0	26	37	27	32.0	-1	2	11
2	1-103	2.0	92.0	3	9	11.0	5.0	14.0	2.0	2.0	...	3.0	3.0	35.0	24	34	27	1.0	-3	2	8
3	1-105	1.0	92.0	3	9	12.0	4.0	12.0	3.0	1.0	...	3.0	3.0	21.0	24	39	36	-18.0	-12	3	31
4	1-106	2.0	92.0	3	9	15.0	3.0	14.0	2.0	3.0	...	4.0	4.0	27.0	38	31	24	-4.0	14	4	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
1002	14-x42	1.0	89.0	2	11	20.0	2.0	14.0	2.0	3.0	...	4.0	4.0	0.0	0	0	0	0.0	0	1	18
1003	14-x43	2.0	89.0	2	11	4.0	5.0	14.0	2.0	3.0	...	3.0	3.0	16.0	24	42	38	-26.0	-14	3	13
1004	14-x44	1.0	89.0	2	11	11.0	4.0	14.0	3.0	2.0	...	4.0	4.0	0.0	0	0	0	0.0	0	1	37
1005	14-x45	1.0	90.0	2	11	1.0	5.0	21.0	5.0	1.0	...	4.0	3.0	17.0	24	44	35	-27.0	-11	3	59
1006	14-x46	1.0	89.0	2	11	16.0	5.0	13.0	4.0	1.0	...	4.0	1.0	14.0	23	46	37	-32.0	-14	3	64

1007 rows × 162 columns

In [12]:
```
df.drop(columns = "Öğrenci No", inplace = True)
```

In [13]:
```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor

df = df.dropna()

X, y = df.drop(columns=["pass", "ist", "Not Ortalaması aps"]), df["Not Ortalaması aps"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=47)

model_linear = LinearRegression()
model_linear.fit(X_train, y_train)

y_pred_linear_test = model_linear.predict(X_test)
r2_linear_test = r2_score(y_test, y_pred_linear_test)
print("Linear Regression Test R^2 Score:", r2_linear_test)

y_pred_linear_train = model_linear.predict(X_train)
r2_linear_train = r2_score(y_train, y_pred_linear_train)
print("Linear Regression Train R^2 Score:", r2_linear_train)
print("-------------------------------------")
model_random_forest = RandomForestRegressor(n_estimators=100, min_samples_split=2)
model_random_forest.fit(X_train, y_train)

y_pred_rf_test = model_random_forest.predict(X_test)
r2_rf_test = r2_score(y_test, y_pred_rf_test)
print("Random Forest Test R^2 Score:", r2_rf_test)

y_pred_rf_train = model_random_forest.predict(X_train)
r2_rf_train = r2_score(y_train, y_pred_rf_train)
print("Random Forest Train R^2 Score:", r2_rf_train)
print("-------------------------------------")

model_gradient_boosting = GradientBoostingRegressor(n_estimators=80, min_samples_split=2)
model_gradient_boosting.fit(X_train, y_train)

y_pred_gb_test = model_gradient_boosting.predict(X_test)
r2_gb_test = r2_score(y_test, y_pred_gb_test)
print("Gradient Boosting Test R^2 Score:", r2_gb_test)

y_pred_gb_train = model_gradient_boosting.predict(X_train)
r2_gb_train = r2_score(y_train, y_pred_gb_train)
print("Gradient Boosting Train R^2 Score:", r2_gb_train)
```

```
Linear Regression Test R^2 Score: 0.9415268586047371
Linear Regression Train R^2 Score: 1.0
-------------------------------------
Random Forest Test R^2 Score: 0.8728767381416505
Random Forest Train R^2 Score: 0.9883898151137558
-------------------------------------
Gradient Boosting Test R^2 Score: 0.9453778398257066
Gradient Boosting Train R^2 Score: 0.9999509933306359
```

In [14]:
```
df = df.dropna()
X, y = df.drop(columns=["pass", "ist", "Not Ortalaması aps"]), df["ist"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=47)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

r2 = r2_score(y_test, y_pred)
print("Linear Reg Test R-squared:", r2)

y_pred_train = model.predict(X_train)
r2_train = r2_score(y_train, y_pred_train)
print("Linear Reg Train R-squared:", r2_train)

print("-------------------------------------")

model_random_forest = RandomForestRegressor(n_estimators=100, min_samples_split=2)
model_random_forest.fit(X_train, y_train)

y_pred = model_random_forest.predict(X_test)

r2 = r2_score(y_test, y_pred)
print("Random Forest Test R^2 Score:", r2)


y_pred_rf_train = model_random_forest.predict(X_train)
r2_rf_train = r2_score(y_train, y_pred_rf_train)
print("Random Forest Train R^2 Score:", r2_rf_train)

print("-------------------------------------")

model = GradientBoostingRegressor(n_estimators=80, min_samples_split=2)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("Gradient Boosting Test R^2 Score:", r2)


y_pred_gb_train = model.predict(X_train)
r2_gb_train = r2_score(y_train, y_pred_gb_train)
print("Gradient Boosting Train R^2 Score:", r2_gb_train)
```

```
Linear Reg Test R-squared: -0.6638937244373464
Linear Reg Train R-squared: 1.0
-------------------------------------
Random Forest Test R^2 Score: 0.9907547008547009
Random Forest Train R^2 Score: 0.9917107211205271
-------------------------------------
Gradient Boosting Test R^2 Score: 0.9909999512073977
Gradient Boosting Train R^2 Score: 0.9999999512688926
```

In [15]:
```
df = df.dropna()
X, y = df.drop(columns=["pass", "ist", "Not Ortalaması aps"]), df["pass"]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=47)

model = LinearRegression()
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

r2 = r2_score(y_test, y_pred)
print("Linear Reg Test R-squared:", r2)

y_pred_train = model.predict(X_train)
r2_train = r2_score(y_train, y_pred_train)
print("Linear Reg Train R-squared:", r2_train)

print("-------------------------------------")

model_random_forest = RandomForestRegressor(n_estimators=100, min_samples_split=2)
model_random_forest.fit(X_train, y_train)

y_pred = model_random_forest.predict(X_test)

r2 = r2_score(y_test, y_pred)
print("Random Forest Test R^2 Score:", r2)


y_pred_rf_train = model_random_forest.predict(X_train)
r2_rf_train = r2_score(y_train, y_pred_rf_train)
print("Random Forest Train R^2 Score:", r2_rf_train)

print("-------------------------------------")

model = GradientBoostingRegressor(n_estimators=80, min_samples_split=2)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)
r2 = r2_score(y_test, y_pred)
print("Gradient Boosting Test R^2 Score:", r2)


y_pred_gb_train = model.predict(X_train)
r2_gb_train = r2_score(y_train, y_pred_gb_train)
print("Gradient Boosting Train R^2 Score:", r2_gb_train)
```

```
Linear Reg Test R-squared: 0.9991979663941473
Linear Reg Train R-squared: 1.0
-------------------------------------
Random Forest Test R^2 Score: 0.8572814191948919
Random Forest Train R^2 Score: 0.9883385364762134

Gradient Boosting Test R^2 Score: 0.9238559682091917
Gradient Boosting Train R^2 Score: 0.9956604276578465
```

In [ ]: