

# Öğrenci Performans Verisi Analizi

İsim Soyisim	Numara
Yusuf Doğu	23181616038
MOHAMMED ABDULHAMID ALBAWENDI	221801616071



iDEAL SINIF?

---



# VERİYE İLK BAKIŞ

---



# Veri Boyutu ve Değişken Türleri

```
=====
● ANALİZİ YAPILAN VERİ SETİ : Portekizce Verisi ●
=====

• VERİ BOYUTLARI
  → SATIR SAYISI: 649
  → SÜTUN SAYISI: 33
  -----

• DEĞİŞKEN (SÜTUN) TURLERİ
school      object
sex         object
age         int64
address     object
famsize     object
Pstatus     object
Medu        int64
Fedu        int64
Mjob        object
Fjob        object
```

## Benzersiz Değerler

```
• EKSİK DEĞERLER
✓ EKSİK DEĞER BULUNAMADI
-----

• BENZERSİZ (UNIQUE) DEĞER SAYISI
school      2
sex         2
age         8
address     2
famsize     2
Pstatus     2
Medu        5
Fedu        5
Mjob        5
Fjob        5
reason      4
```

# Örnek Satırlar

```
• VERİDEN KARIŞIK ŞEKİLDE 5 ÖRNEK
```

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	reason	guardian	traveltime	studytime	failures	schoolsup	famsup	paid
370	GP	F	19	U	LE3	A	1	1	other	other	course	other	3	2	2	no	yes	no
338	GP	F	17	R	LE3	T	3	1	services	other	reputation	mother	2	4	0	no	yes	no
232	GP	F	17	U	GT3	T	2	3	at_home	other	home	father	2	1	0	no	yes	no
281	GP	M	16	U	GT3	T	0	2	other	other	other	mother	1	1	0	no	no	no
132	GP	F	17	U	LE3	A	2	1	other	other	course	mother	3	1	0	no	yes	no
313	GP	F	18	U	LE3	T	1	1	other	other	home	mother	2	2	0	no	yes	no
487	MS	F	18	R	LE3	A	3	2	other	other	course	other	2	3	2	no	yes	no

## İstatistikler

```
• VERİNİN İSTATİSTİK ÖZETİ
```

	count	mean	std	min	0%	5%	50%	95%	99%	100%	max
age	649.0	16.744222	1.218138	15.0	15.0	15.0	17.0	19.0	20.0	22.0	22.0
Medu	649.0	2.514638	1.134552	0.0	0.0	1.0	2.0	4.0	4.0	4.0	4.0
Fedu	649.0	2.306626	1.099931	0.0	0.0	1.0	2.0	4.0	4.0	4.0	4.0
traveltime	649.0	1.568567	0.748660	1.0	1.0	1.0	1.0	3.0	4.0	4.0	4.0
studytime	649.0	1.930663	0.829510	1.0	1.0	1.0	2.0	4.0	4.0	4.0	4.0
failures	649.0	0.221880	0.593235	0.0	0.0	0.0	0.0	1.0	3.0	3.0	3.0
famrel	649.0	3.930663	0.955717	1.0	1.0	2.0	4.0	5.0	5.0	5.0	5.0
freetime	649.0	3.180277	1.051093	1.0	1.0	1.0	3.0	5.0	5.0	5.0	5.0
goout	649.0	3.184900	1.175766	1.0	1.0	1.0	3.0	5.0	5.0	5.0	5.0

# KATEGORİK Mİ SAYISAL MI ?

```
def kat_num_analiz(df,df_isim,yazdir=True): 2 usages
# Kategorik sütunları belirleme
kat_sutunlar = [col for col in df.columns if df[col].dtypes in ["category", "object"]]

# Sayısal sütunları belirleme
say_sutunlar = [col for col in df.columns if df[col].dtypes in ["int64", "float"]]

if yazdir:
    print("\n" + "=" * 150)
    print(f"✦ {df_isim} VERİSİ SÜTUNLAR İÇİN TÜR ANALİZİ\n".center(150))
    print("=" * 150 + "\n")

    # Genel bilgiler
    print(f"♦ Toplam Sütun Sayısı: {df.shape[1]}")
    print(f"♦ Kategorik Sütun Sayısı: {len(kat_sutunlar)}")
    print(f"♦ Sayısal Sütun Sayısı: {len(say_sutunlar)}")

    # Unique değer sayılarının analizi
    print(f"📊 Kategorik Sütunların Unique Değer Sayıları:")
    print(df[kat_sutunlar].nunique(), "\n")

    print(f"📊 Sayısal Sütunların Unique Değer Sayıları:")
    print(df[say_sutunlar].nunique(), "\n")

return kat_sutunlar, say_sutunlar
```

```
=====
✦ Matematik Verisi VERİSİ SÜTUNLAR İÇİN TÜR ANALİZİ
=====

♦ Toplam Sütun Sayısı: 33
♦ Kategorik Sütun Sayısı: 17
♦ Sayısal Sütun Sayısı: 16
📊 Kategorik Sütunların Unique Değer Sayıları:
school      2
sex         2
address     2
famsize     2
Pstatus     2
Mjob       5
Fjob       5
reason     4
```

# KATEGORİK GÖRSELLEŞTİRME

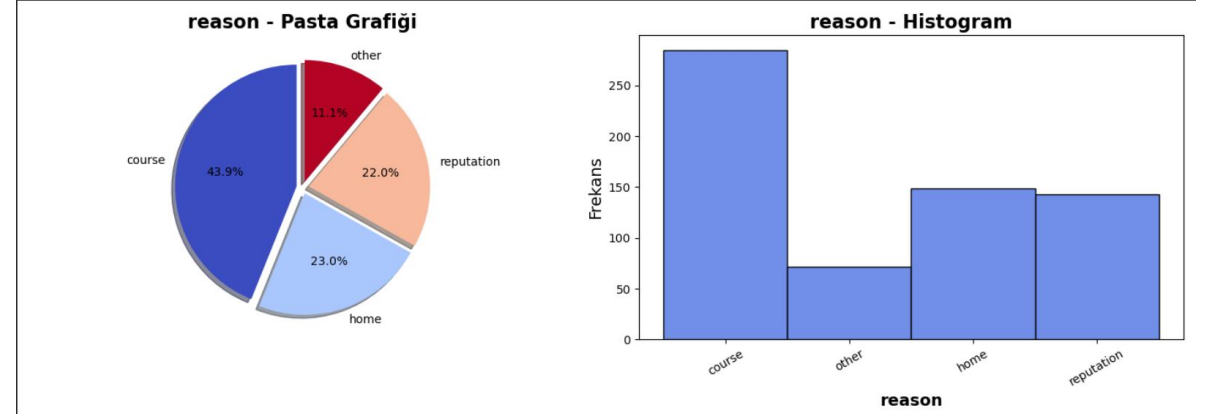
```
def kat_bool_ozet(df, df_isim, kat_sutunlar, sinif_limiti=10): 2 usages
    print(f"\n{'=' * 200}")
    print(f"📊 {df_isim.upper()} VERİ SETİ KATEGORİK ANALİZİ".center(200))
    print(f"\n{'=' * 200}\n")
    for kat_sutun in kat_sutunlar:
        benzersiz_sayilar = df[kat_sutun].nunique()
        print(f"\n{'-' * 120}")
        print(f"{kat_sutun.upper()} DEĞİŞKENİ ANALİZİ (Unique Değerler: {benzersiz_sayilar})".center(120))
        print(f"\n{'-' * 120}\n")

        value_counts = df[kat_sutun].value_counts(normalize=True, dropna=False) * 100
        fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(14, 5))

        # Pasta Grafiği
        value_counts.plot.pie(
            autopct="%.1f%%", startangle=90, cmap="coolwarm", shadow=True,
            explode=[0.05] * len(value_counts), ax=axes[0]
        )
        axes[0].set_ylabel("")
        axes[0].set_title(f"{kat_sutun} - Pasta Grafiği", fontsize=16, fontweight="bold")

        # Histogram
        sns.histplot(df[kat_sutun], bins=min(20, benzersiz_sayilar), kde=False, ax=axes[1],
                     color="royalblue")
        axes[1].set_xlabel(kat_sutun, fontsize=14, fontweight="bold")
        axes[1].set_ylabel("Frekans", fontsize=14)
        axes[1].set_title(f"{kat_sutun} - Histogram", fontsize=16, fontweight="bold")
        axes[1].tick_params(axis="x", rotation=30, labelsize=10)

    plt.tight_layout()
    plt.show()
```



# SAYISAL GÖRSELLEŐTİRME

```
def num_ozet(df, df_isim, say_sutunlar): 2 usages
    print(f"\n{'=' * 200}")
    print(f"📊 {df_isim.upper()} VERİ SETİ SAYISAL ANALİZİ".center(200))
    print(f"{'=' * 200}\n")

    if not say_sutunlar:
        print(f"⚠️ {df_isim} veri setinde sayısal sütun bulunmamaktadır.")
        return

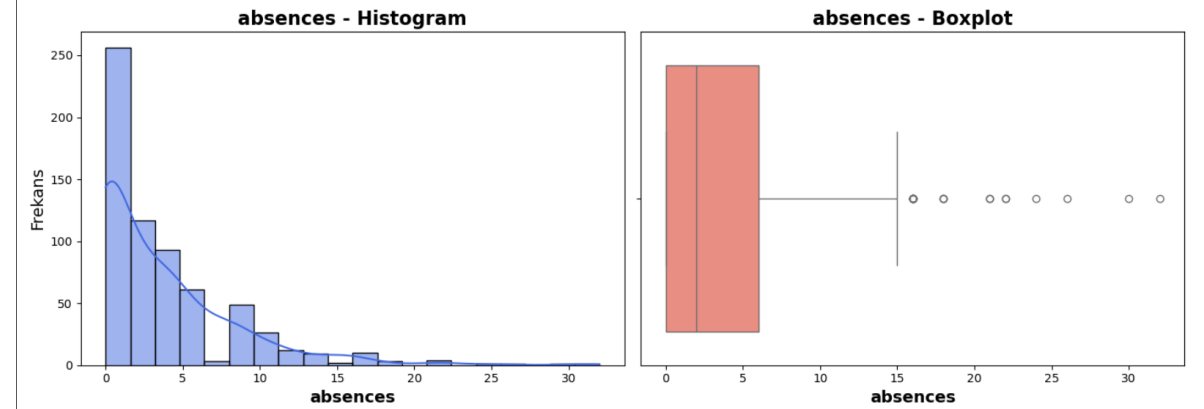
    for say_sutun in say_sutunlar:
        print(f"\n{'-' * 120}")
        print(f"{say_sutun.upper()} DEĞİŐKEMİ ANALİZİ".center(120))
        print(f"{'-' * 120}\n")

        fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(14, 5))

        # Histogram
        sns.histplot(df[say_sutun], bins=20, kde=True, ax=axes[0], color="royalblue")
        axes[0].set_xlabel(say_sutun, fontsize=14, fontweight="bold")
        axes[0].set_ylabel("Frekans", fontsize=14)
        axes[0].set_title(f"{say_sutun} - Histogram", fontsize=16, fontweight="bold")

        # Boxplot
        sns.boxplot(x=df[say_sutun], ax=axes[1], color="salmon")
        axes[1].set_xlabel(say_sutun, fontsize=14, fontweight="bold")
        axes[1].set_title(f"{say_sutun} - Boxplot", fontsize=16, fontweight="bold")

    plt.tight_layout()
    plt.show()
```



# KORELASYON ANALİZİ

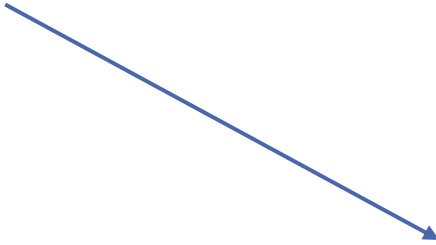
```
def sicaklik_haritasi(df,df_isim): 2 usages
    corr = df.corr()
    g1 = corr[abs(corr['G1']) >= 0.1]['G1'].drop('G1').sort_values(ascending=False)
    g2 = corr[abs(corr['G2']) >= 0.1]['G2'].drop('G2').sort_values(ascending=False)
    g3 = corr[abs(corr['G3']) >= 0.1]['G3'].drop('G3').sort_values(ascending=False)
    print(g1, g2, g3)
    list_x=[g1,g2,g3]
    i=1
    for x in list_x:
        plt.figure(figsize=(20, 18))
        sns.heatmap(x.to_frame(), cmap='coolwarm', annot=True)
        plt.title(f'{df_isim} G{i} için sıcaklık haritası')
        plt.savefig(f'{df_isim}{i}.png')
        plt.show()
        i+=1

sicaklik_haritasi(onehot_mat_df, df_isim: 'Matematik Verisi ')
sicaklik_haritasi(onehot_por_df, df_isim: 'Portekiz Verisi ')
```



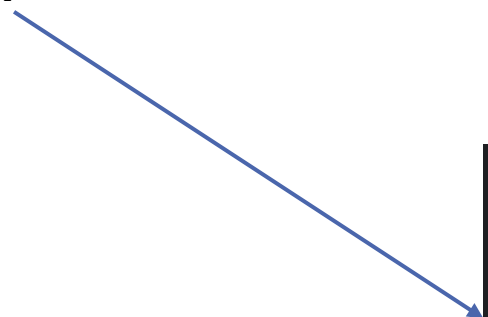


**MATEMATİK VERİSİ İÇİN**  
**33 SÜTUNDAN 18**  
**SÜTUNA**



```
mat_selected_features = [  
    "failures", "Medu", "Fedu", "higher", "romantic", "Mjob", "Fjob",  
    "address", "sex", "goout", "traveltime", "age", "studytime",  
    "Walc", "schoolsup", "internet", "paid", "Basari"  
]
```

**PORTEKİZCE VERİSİ İÇİN**  
**33 SÜTUNDAN 20**  
**SÜTUNA**



```
por_selected_features = [  
    "failures", "school", "Medu", "Fedu", "higher", "Mjob", "Fjob",  
    "address", "sex", "traveltime", "age", "studytime", "reason", "guardian",  
    "Walc", "Dalc", "internet", "absences", "freetime", "Basari"  
]
```

# SON AŞAMA

```
def one_hot_encoder(df): 4 usages
    # Kategorik sütunları seçelim
    kat_sutunlar = df.select_dtypes(include="object").columns

    # Encoder
    encoder = OneHotEncoder(sparse_output=False)

    # Encode işlemi ve yeni DataFrame
    encoded_df = pd.DataFrame(encoder.fit_transform(df[kat_sutunlar]),
                              columns=encoder.get_feature_names_out(kat_sutunlar))

    encoded_df.head()
    yeni_df=df.drop(columns=kat_sutunlar)
    # Eski kategorik sütunları atıp encode edilmiş sütunları ekleyelim
    yeni_df = yeni_df.join(encoded_df)

    return yeni_df
```

OneHotEncoding ile Sayısallaştırma

```
def olceklendirme(df): 4 usages

    # Sayısal sütunları seçiyoruz
    numerical_cols = df.select_dtypes(include=['float64', 'int64']).columns

    scaler = MinMaxScaler()

    df[numerical_cols] = scaler.fit_transform(df[numerical_cols])
    return df

scaled18_binary_target_mat_df=olceklendirme(binary18_target_mat_df)
scaled33_binary_target_mat_df=olceklendirme(binary33_target_mat_df)

scaled20_binary_target_por_df=olceklendirme(binary20_target_por_df)
scaled33_binary_target_por_df=olceklendirme(binary33_target_por_df)
```

MinMaxScaler ile 0-1 Arasına Ölçeklendirme

# VERİNİN YENİ HALİ

[illegible]

Matematik Verisi	Portekizce Verisi
33 sütunlu ham veri	33 sütunlu ham veri
18 sütunlu düzenlenmiş veri	20 sütunlu düzenlenmiş veri

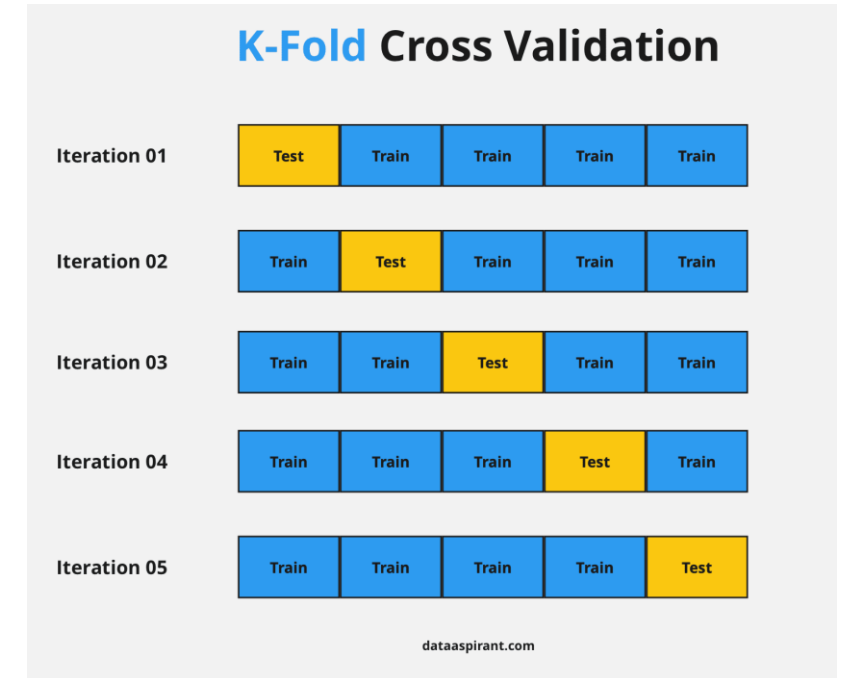


# Model Doğrulama Teknikleri: K-Fold & GridSearch

## 1- K-Fold Çapraz Doğrulama (K-Fold Cross Validation):

Modelin doğruluğunu ve güvenilirliğini daha gerçekçi bir şekilde değerlendirmek için kullanılan bir tekniktir.

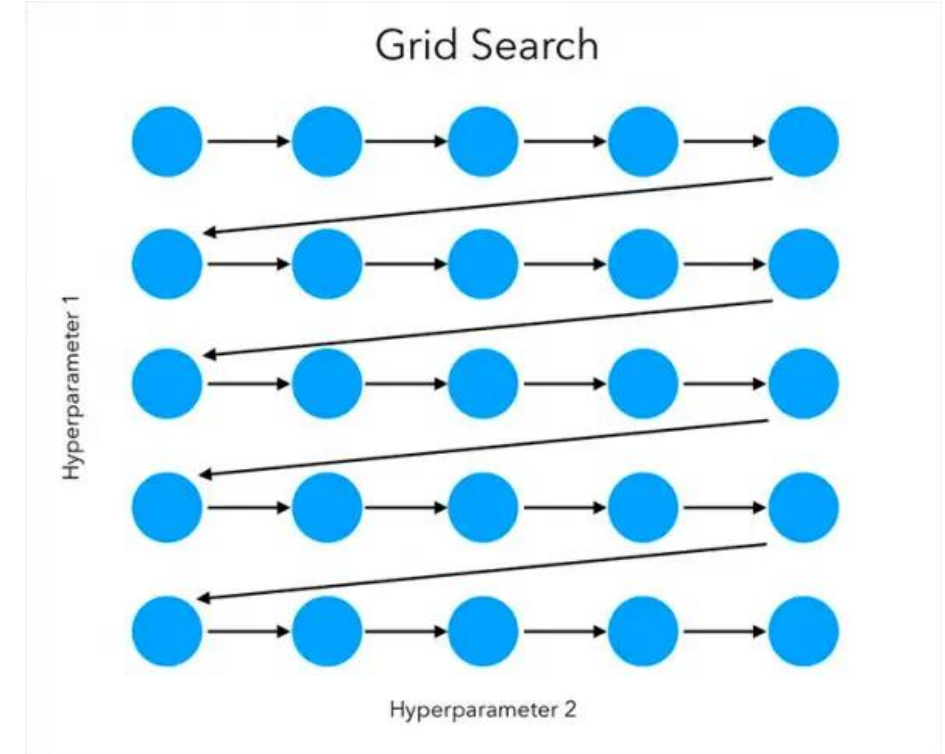
- Veri kümesi K eşit parçaya (Fold) bölünür.
- Her seferinde bir parça test için, geri kalanı eğitim için kullanılır.
- Bu işlem K kez tekrarlanır, her seferinde farklı bir parça test verisi olur.
- Sonuçların ortalaması alınarak genel bir başarı puanı elde edilir.



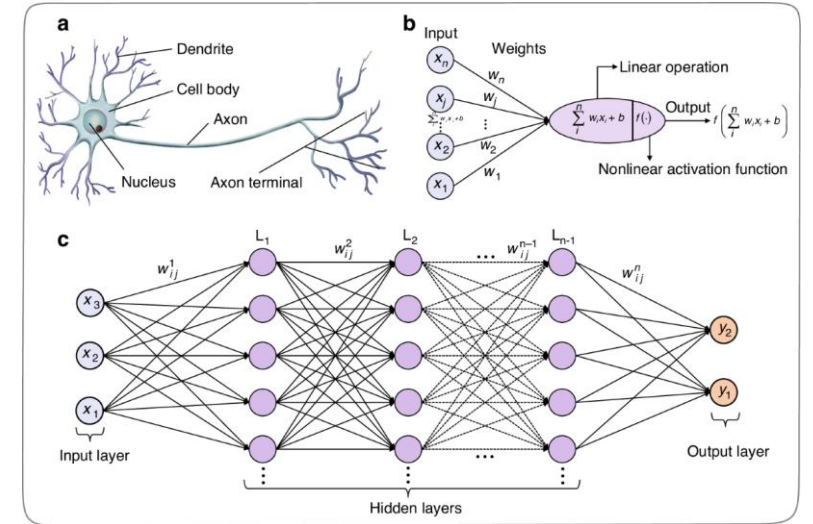
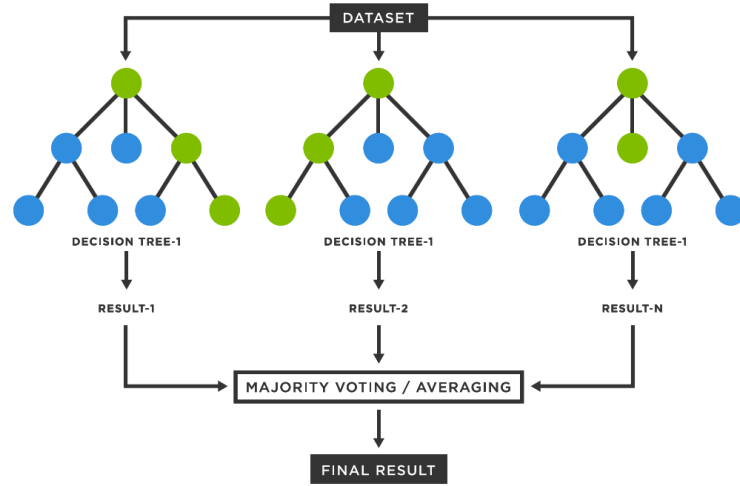
## 2- GridSearch:

Modelin performansını en üst düzeye çıkaracak en iyi hiperparametre kombinasyonlarını bulmak için kullanılan bir yöntemdir.

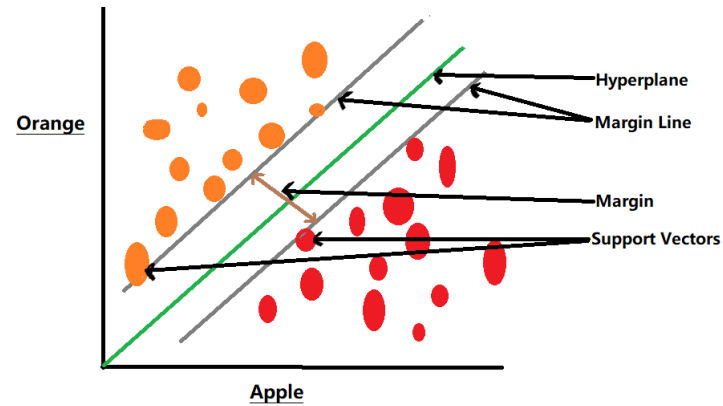
- Her hiperparametre için olası değer aralıkları belirlenir.
- Tüm olası kombinasyonlar denenir.
- Her kombinasyon, K-Fold çapraz doğrulama ile test edilir.
- En yüksek başarıyı sağlayan kombinasyon seçilir.



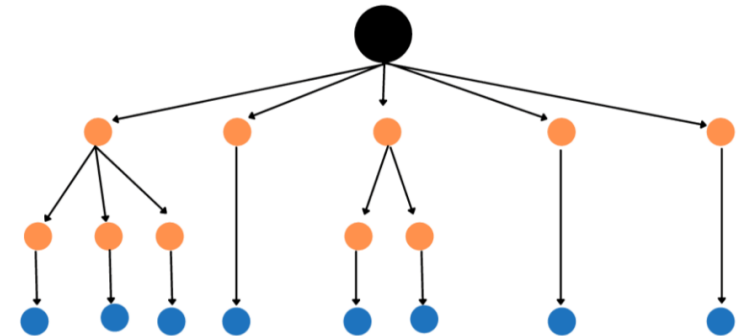
# Makine Öğrenmesi Modelleri



## Support Vector Machines



## Decision Tree





# 1- Karar Ağaçları(Decision Trees):

Decision Tree, veriyi dallara ayırarak kararlar alan ve sınıflandırma yapan basit ama etkili bir makine öğrenmesi algoritmasıdır.

Her bir düğüm, bir özelliğe göre veri setini ayırır; yaprak düğümler ise nihai sınıflandırmayı temsil eder.

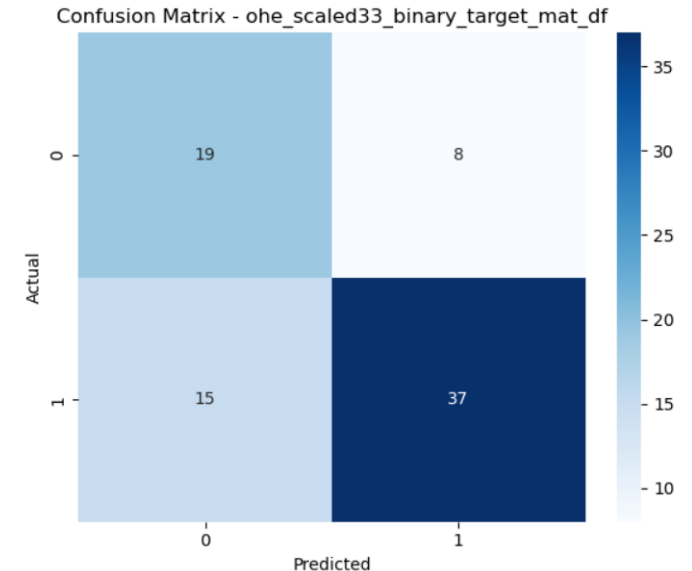
Yorumlaması kolaydır ve görselleştirilebilir olması sayesinde eğitim alanında sıkça tercih edilir.

## Nasıl uyguladık?

- 4 farklı veri kümesi (2 Matematik + 2 Portekizce) üzerinde uygulandı.
- Veri, eğitim (%80) ve test (%20) olarak ayrıldı.
- GridSearchCV kullanılarak en iyi hiperparametreler arandı: (criterion, max\_depth, min\_samples\_split, min\_samples\_leaf)
- Her veri kümesi için en iyi sonucu veren parametreler seçildi.
- Model doğruluğu test verisi ile ölçüldü ve confusion matrix ile görselleştirildi.

```
# Grid Search
grid_search = GridSearchCV(DecisionTreeClassifier(random_state=42), param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)
best_params = grid_search.best_params_
print(f" Best Parameters: {best_params}")

# En iyi model
model = DecisionTreeClassifier(**best_params, random_state=42)
model.fit(X_train, y_train)
```



## 2- Rastgele Ormanlar(Random Forests):

Random Forest, birden fazla karar ağacının birlikte çalıştığı bir makine öğrenmesi algoritmasıdır. Her ağaç farklı veri örnekleriyle eğitilir ve nihai tahmin tüm ağaçların oylarıyla belirlenir. Bu yapı, hem doğruluğu artırır hem de overfitting riskini azaltır.

### Nasıl Uyguladık ?

- 4 farklı veri kümesi üzerinde uyguladık.
- K-Fold (5 katlı) çapraz doğrulama ile model sağlamlığı artırıldı.
- GridSearchCV ile en iyi parametreler arandı:  
(n\_estimators, max\_depth, min\_samples\_split, min\_samples\_leaf)
- Her veri kümesinde en yüksek başarı veren model seçildi.
- Random Forest, projede en iyi sonuçlardan bazılarını verdi.

```
kf = KFold(n_splits=5, shuffle=True, random_state=42)

for fold, (train_idx, test_idx) in enumerate(kf.split((constant) X: DataFrame)):
    X_train, X_test = X.iloc[train_idx], X.iloc[test_idx]
    y_train, y_test = y.iloc[train_idx], y.iloc[test_idx]

    grid_search = GridSearchCV(RandomForestClassifier(random_state=42), param_grid, cv=5, scoring='accuracy', n_jobs=-1)
    grid_search.fit(X_train, y_train)
    best_params = grid_search.best_params_

    model = RandomForestClassifier(**best_params, random_state=42)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
```

### 3- Destek Vektör Makinesi(SVM):

SVM, veriyi sınıflandırmak için en uygun ayırıcı çizgiyi (veya düzlemi) bulmaya çalışan güçlü bir makine öğrenmesi algoritmasıdır.

Özellikle yüksek boyutlu ve karmaşık veri kümelerinde etkili sonuçlar verir.

### Nasıl uyguladık?

- 4 farklı veri kümesi üzerinde uyguladık (2 Matematik + 2 Portekizce).
- K-Fold (5 katlı) çapraz doğrulama ile modelin kararlılığı test edildi.
- GridSearchCV ile en iyi parametreler arandı:  
(C, kernel, gamma)
- Her kombinasyon 3 katlı doğrulama ile değerlendirildi (cv=3).
- En yüksek doğruluğu veren model her veri kümesi için seçildi.
- SVM, bazı veri kümelerinde yüksek doğruluk sağladı.

```
kf = KFold(n_splits=5, shuffle=True, random_state=42)

param_grid = {
    'C': [0.1, 1, 10],
    'kernel': ['linear', 'rbf'],
    'gamma': ['scale', 'auto']
}

grid_search = GridSearchCV(SVC(), param_grid, cv=3, scoring='accuracy', n_jobs=-1)
grid_search.fit(X_train, y_train)

best_svm = grid_search.best_estimator_
y_pred = best_svm.predict(X_test)
```



## 4- Yapay Sinir Ağları(ANN) :

ANN, insan beynindeki nöron yapısından esinlenerek geliştirilen bir makine öğrenmesi yöntemidir. Çok katmanlı yapı sayesinde karmaşık ve doğrusal olmayan ilişkileri öğrenebilir. Eğitim verisinden örüntüleri öğrenerek tahmin yapabilir.

### Nasıl uyguladık?

- 4 farklı veri kümesinde uygulandı.
- Her biri için 5 katlı K-Fold çapraz doğrulama yapıldı.
- Model Keras kullanılarak oluşturuldu:
  - Giriş katmanı: input\_dim=X.shape[1]
  - Gizli katmanlar: 64 ve 32 nöron, ReLU aktivasyonu
  - Çıkış katmanı: 1 nöron, sigmoid aktivasyonu
- 50 epoch ve batch\_size=32 ile eğitim yapıldı.
- En yüksek doğruluk sağlayan model seçildi ve testte değerlendirildi.

```
model = Sequential()  
model.add(Dense(64, activation='relu', input_dim=X_train.shape[1]))  
model.add(Dense(32, activation='relu'))  
model.add(Dense(1, activation='sigmoid'))  
  
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])  
model.fit(X_train, y_train, epochs=50, batch_size=32, verbose=0)
```

# MODEL SONUÇLARI

Model	MATEMATİK VERİSİ (18 sütun)	MATEMATİK VERİSİ (33 sütun)	PORTEKİZCE VERİSİ (20 sütun)	PORTEKİZCE VERİSİ (33 sütun)
Random Forest	73.42%	<u>74.68%</u>	<u>84.50%</u>	82.36%
Decision Tree	<u>69.62%</u>	68.35%	78.46%	<u>79.75%</u>
SVM	<u>78.48%</u>	70.89%	<u>81.40%</u>	80.50%
YSA	72.15%	<u>75.95%</u>	<u>81.54%</u>	81.40%

# SONUÇLAR VE ÖNERİLER

**Eğitim  
nedir ?**

**Eğitimin amacı  
nedir ?**

**Eğitim düzeyi sadece notlar ile  
mi ölçülür ?**

## Öneriler:

- 1- İleride yapılacak çalışmalarda, sadece not ve demografik veriler değil; psikolojik, motivasyonel ve duygusal zekâ gibi nitel veriler de modele dâhil edilerek daha kapsayıcı bir başarı tahmin sistemi kurulabilir.
- 2- Veri setine zaman serisi analizi gibi yaklaşımlar uygulanarak, öğrencilerin başarı trendleri dinamik olarak izlenebilir ve bu veriler, öğretmen-veli iş birliğini güçlendirecek geri bildirim sistemlerinde kullanılabilir.
- 3- Bu veriye ek olarak öğretmen niteliği ile ilgili değişkenler katılmalıdır , sınavdaki zorluk düzeyi dersi anlatma şekli bunların hepsi önemli faktörler olabilir



***Vatanını En Çok Seven Görevini En İyi  
Yapandır***

***Başöğretmen Gazi Mustafa Kemal Atatürk***