

# Lojistik Regresyon

NOT: Lojistik Regresyon tahmin sonuçlarını [0, 1] aralığında sınırlar ve sınıflandırma problemleri için kullanılırken, Doğrusal Regresyon tahmin sonuçları sürekli değerlerdir ve regresyon problemleri için kullanılır.

## Lojistik Regresyon ( Logistic Regression )

Amaç sınıflandırma problemi için bağımlı ve bağımsız değişkenler arasındaki ilişkiyi doğrusal olarak modellemektir.

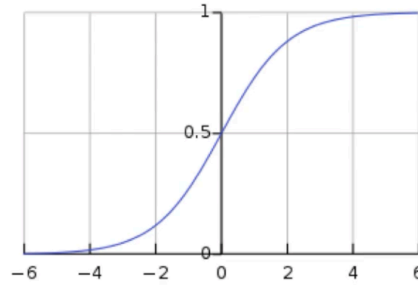
$$p(\hat{y}_i) = \frac{1}{1 + e^{-(z)}} \quad \odot$$
$$\rightarrow z = b + w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_px_p$$

Nasıl? Gerçek değerler ile tahmin edilen değerler arasındaki farklara ilişkin log loss değerini minimum yapabilecek ağırlıkları bularak.

$$\hat{y}_i = \frac{1}{1 + e^{-(z)}} \quad \odot \quad \downarrow \text{MSE}$$
$$z = b + w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_px_p$$
$$\text{Log Loss} = \frac{1}{m} \left( \sum_{i=1}^m -y_i * \log(p(\hat{y}_i)) - (1 - y_i) \log(1 - p(\hat{y}_i)) \right)$$

### Sigmoid Function:

$$\hat{y}_i = P(Y = 1 | X = x)$$



$$\frac{1}{1 + e^{-(z)}}$$

$$z = b + w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_px_p$$

## Nested Logistic regression → Deep Learning:

Soru: Verilen bias ve weight'lere göre aşağıdaki gözlem birimi için 1 sınıfına ait olma olasılığını hesaplayınız.

$$b = 5, w_1 = 4, w_2 = -4, w_3 = 3$$

$$x_1 = 2, x_2 = 3, x_3 = 0$$

İpucu:  $\frac{1}{1 + e^{-(z)}}$   $z = b + w_1x_1 + w_2x_2 + w_3x_3 + \dots + w_px_p$

Çözüm:

$$z = 5 + (4) * 2 + (-4) * 3 + 3 * (0)$$

$$z = 1$$

$$\frac{1}{1 + e^{-(1)}} = 0.731$$

## Lojistik Regresyon için Gradient Descent

Cross Entropy: Entropy düşük çeşitlilik az olması istenir

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)})$$

↓ 0.10 ↑  
0.30

$$J(\theta) = \frac{1}{m} \left[ \sum_{i=1}^m -y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})) \right]$$

-1 log(0.80)      0.0

$$J(\theta) = \frac{1}{m} \sum_{i=1}^m \text{Cost}(h_{\theta}(x^{(i)}), y^{(i)})$$

$$J(\theta) = \frac{1}{m} \left[ \sum_{i=1}^m -y^{(i)} \log(h_{\theta}(x^{(i)})) + (1 - y^{(i)}) \log(1 - h_{\theta}(x^{(i)})) \right]$$



Repeat {

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta)$$

}

## Sınıflandırma Problemlerinde Başarı Değerlendirme

## churn

Age	Total_Purchase	Years	Churn	Predicted_Churn
42.0	11066.8	7.22	1	1
41.0	11916.22	6.5	1	0
38.0	12884.75	6.67	0	0
42.0	8010.76	6.71	1	0
37.0	9191.58	5.56	0	1
48.0	10356.02	5.12	1	1
44.0	11331.58	5.23	0	0
32.0	9885.12	6.92	0	0
43.0	14062.6	5.46	1	1
40.0	8066.94	7.11	1	1

### Confusion Matrix:

		Tahmin Edilen Sınıf	
		Sınıf = 1	Sınıf = 0
Gerçek Sınıf	Sınıf = 1	True Pozitif (TP)	False Negatif (FN)
	Sınıf = 0	False Pozitif (FP)	True Negatif (TN)

**Accuracy:** Doğru sınıflandırma oranıdır:  $(TP+TN) / (TP+TN+FP+FN)$

**Precision:** Pozitif sınıf (1) tahminlerinin başarı oranıdır:  $TP / (TP+FP)$

**Recall:** Pozitif sınıfın (1) doğru tahmin edilme oranıdır:  $TP / (TP + FN)$


**F1 SCORE:**  $2 * (Precision * Recall) / (Precision + Recall)$

# Karmaşıklık Matrisi ( Confusion Matrix )

Soru: 1000 kredi kartı işlemi var. 990 normal işlem. 10 sahtekar işlem.

Buna göre Confusion matrisi doldurunuz ve başarı metriklerini hesaplayınız.

		Tahmin Edilen Sınıf Değerleri		
		Fraud İşlem (1)	Normal İşlem (0)	
Gerçek Sınıf Değerleri	Fraud İşlem (1)	5	5	<b>10</b>
	Normal İşlem (0)	90	900	<b>990</b>
		<b>95</b>	<b>905</b>	

- Accuracy =  $(5 + 900) / 1000 = 0.905$  
- Precision =  $5 / (5+90) = 0.05$
- Recall =  $5 / (5+5) = 0.50$
- F1 Score =  $2 * 0.025 / 0.55 = 0.09$  0.09

## Classification Threshold

Churn	Predicted Churn	Probability of Class 1
1	1	0,80 ✓
1	0	0,48
0	0	0,30
1	0	0,45
0	1	0,55
1	1	0,70
0	0	0,42
0	0	0,35
1	1	0,60 ✓
1	1	0,70 ✓

$$\text{Accuracy} = \frac{\text{Doğru Sınıflandırma Sayısı}}{\text{Toplam Sınıflandırılan Gözlem Sayısı}}$$

Classification threshold = 0.50 ise accuracy nedir?

$$7/10 = 0.70$$

Churn	Predicted_Churn_0.75	Probability of Class 1
1	1	0,80
1	0	0,48
0	0	0,30
1	0	0,45
0	0	0,55
1	0	0,70
0	0	0,42
0	0	0,35
1	0	0,60
1	0	0,70

$$\text{Accuracy} = \frac{\text{Doğru Sınıflandırma Sayısı}}{\text{Toplam Sınıflandırılan Gözlem Sayısı}}$$

Classification threshold = 0.75 ise accuracy nedir?

$$5/10 = 0.50$$

Churn	Predicted_Churn_0.40	Probability of Class 1
1	1 ✓	0,80
1	1 ✓	0,48
0	0 ✓	0,30
1	1 ✓	0,45
0	1 ✓	0,55
1	1	0,70
0	1	0,42
0	0	0,35
1	1	0,60
1	1	0,70

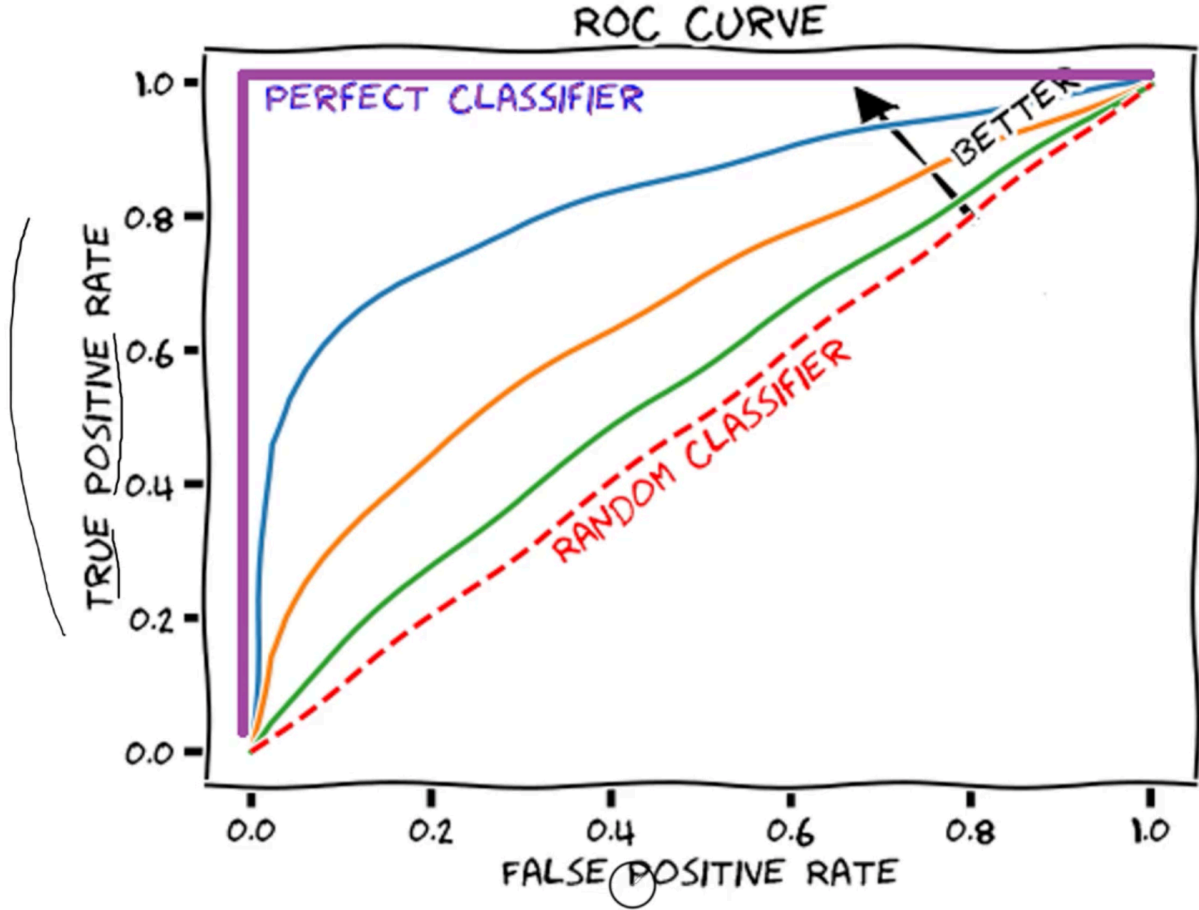
$$\text{Accuracy} = \frac{\text{Doğru Sınıflandırma Sayısı}}{\text{Toplam Sınıflandırılan Gözlem Sayısı}}$$

Classification threshold = 0.40 ise accuracy nedir?

$$8/10 = 0.80$$

- Threshold yukarı çekince Accuracy düştü (Genelde)
- Eşik değeri değişimi?

## ROC Eğrisi ( ROC Curve )



- **ROC Eğrisi** ve **AUC**, bu eşik değerinin değiştirilmesinin model performansını nasıl etkilediğini anlamamızı sağlar.
- AUC, tüm olası sınıflandırma eşikleri için toplu bir performans ölçüsüdür. **AUC (ROC Eğrisi Altındaki Alan)**, ROC eğrisinin altında kalan tüm alandır. ROC eğrisi, modelin çeşitli eşik değerlerindeki performansını gösterirken, AUC bu performansın **tek bir sayısal özetidir**.

## LOG Loss

Cross Entropy Metriği:

- Entropi → Bilgi ve çeşitliliktir

Cross Entropy'nin kökeni Bilgi Teorisi'ne dayanır.

Bilgi teorisinde, entropi bir olayın **belirsizliğini** veya **rastgeleliğini** ölçer. Bir mesajı kodlamak için gereken minimum bit sayısını da düşünebilirsiniz.

- **Entropi (H(P))**: Bir rastgele değişkenin (örneğin bir sınıf etiketi) olasılık dağılımının içerdiği "bilgi miktarının" veya "belirsizliğin" ölçüsüdür.
- **Kullback-Leibler (KL) Divergence (DKL(P || Q))**: İki olasılık dağılımı (P ve Q) arasındaki farkı ölçer. Ne kadar farklı olduklarını, yani Q dağılımının P dağılımından sapmasının ne kadar "bilgi kaybına" yol açtığını gösterir.

Cross Entropy, KL Divergence ile yakından ilişkilidir ve genellikle sınıflandırma modellerinde kullanılan bir **kayıp fonksiyonu** olarak karşımıza çıkar.

### Tanım ve Amaç

**Cross Entropy**, modelin tahmin ettiği olasılık dağılımı (Q) ile olayların gerçek olasılık dağılımı (P) arasındaki farkı ölçer. Daha spesifik olarak, gerçek dağılımın P olduğu varsayılırken, olayları Q dağılımına göre kodlamanın maliyetidir.

Amacımız, modelimizin tahmin ettiği olasılıkların (Q) gerçek etiketlerin olasılıklarına (P) olabildiğince yakın olmasını sağlamaktır. Cross Entropy'yi minimize etmek, modelin tahmin ettiği dağılımın gerçek dağılıma en yakın hale gelmesi anlamına gelir.

### Nasıl Hesaplanır?

Cross Entropy'nin iki ana formu vardır: İkili Sınıflandırma (Binary Cross Entropy) ve Çoklu Sınıflandırma (Categorical Cross Entropy).

Churn	Predicted Churn	Probability of Class 1
→ 1	1	0,80
→ 1	0	0,48
0	0	0,30
1	0	0,45
0	1	0,55
1	1	0,70
0	0	0,42
0	0	0,35
1	1	0,60
1	1	0,70

$$Log Loss = \frac{1}{m} \left( \sum_{i=1}^m -y_i * \log(p(\hat{y}_i)) - (1 - y_i) \log(1 - p(\hat{y}_i)) \right)$$

$- 1 \times \log(0,80)$   
 $- 1 \times \log(0,48)$

## Problem Veri Seti Hikayesi



# İş Problemi:

# Özellikleri belirtildiğinde kişilerin diyabet hastası olup  
# olmadıklarını tahmin edebilecek bir makine öğrenmesi  
# modeli geliştirebilir misiniz?

# Veri seti ABD'deki Ulusal Diyabet-Sindirim-Böbrek Hastalıkları Enstitüler  
i'nde tutulan büyük veri setinin  
# parçasıdır. ABD'deki Arizona Eyaleti'nin en büyük 5. şehri olan Phoenix şe  
hrinde yaşayan 21 yaş ve üzerinde olan  
# Pima Indian kadınları üzerinde yapılan diyabet araştırması için kullanılan v  
erilerdir. 768 gözlem ve 8 sayısal  
# bağımsız değişkenden oluşmaktadır. Hedef değişken "outcome" olarak b  
elirtilmiş olup; 1 diyabet test sonucunun  
# pozitif oluşunu, 0 ise negatif oluşunu belirtmektedir.

# Değişkenler

# Pregnancies: Hamilelik sayısı

# Glucose: Glikoz.

# BloodPressure: Kan basıncı.

# SkinThickness: Cilt Kalınlığı

# Insulin: İnsülin.

# BMI: Beden kitle indeksi.

# DiabetesPedigreeFunction: Soyumuzdaki kişilere göre diyabet olma ihtim  
alimizi hesaplayan bir fonksiyon.

# Age: Yaş (yıl)

# Outcome: Kişinin diyabet olup olmadığı bilgisi. Hastalığa sahip (1) ya da d  
eğil (0)

```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
```

```
from sklearn.preprocessing import RobustScaler
from sklearn.linear_model import LogisticRegression
```

```

#from sklearn.metrics import accuracy_score, roc_auc_score, confusion_m
atrix, classification_report, plot_roc_curve
from sklearn.metrics import accuracy_score, roc_auc_score, confusion_mat
rix, classification_report, RocCurveDisplay

from sklearn.model_selection import train_test_split, cross_validate

def outlier_thresholds(dataframe, col_name, q1=0.05, q3=0.95):
    quartile1 = dataframe[col_name].quantile(q1)
    quartile3 = dataframe[col_name].quantile(q3)
    interquartile_range = quartile3 - quartile1
    up_limit = quartile3 + 1.5 * interquartile_range
    low_limit = quartile1 - 1.5 * interquartile_range
    return low_limit, up_limit

def check_outlier(dataframe, col_name):
    low_limit, up_limit = outlier_thresholds(dataframe, col_name)
    if dataframe[(dataframe[col_name] > up_limit) | (dataframe[col_name] < l
ow_limit)].any(axis=None):
        return True
    else:
        return False

def replace_with_thresholds(dataframe, variable):
    low_limit, up_limit = outlier_thresholds(dataframe, variable)
    dataframe.loc[(dataframe[variable] < low_limit), variable] = low_limit
    dataframe.loc[(dataframe[variable] > up_limit), variable] = up_limit

pd.set_option('display.max_columns', None)
pd.set_option('display.float_format', lambda x: '%.3f' % x)
pd.set_option('display.width', 500)

```

## Keşifçi Veri Analizi ( EDA )

```

#####
# Exploratory Data Analysis

```

```
#####

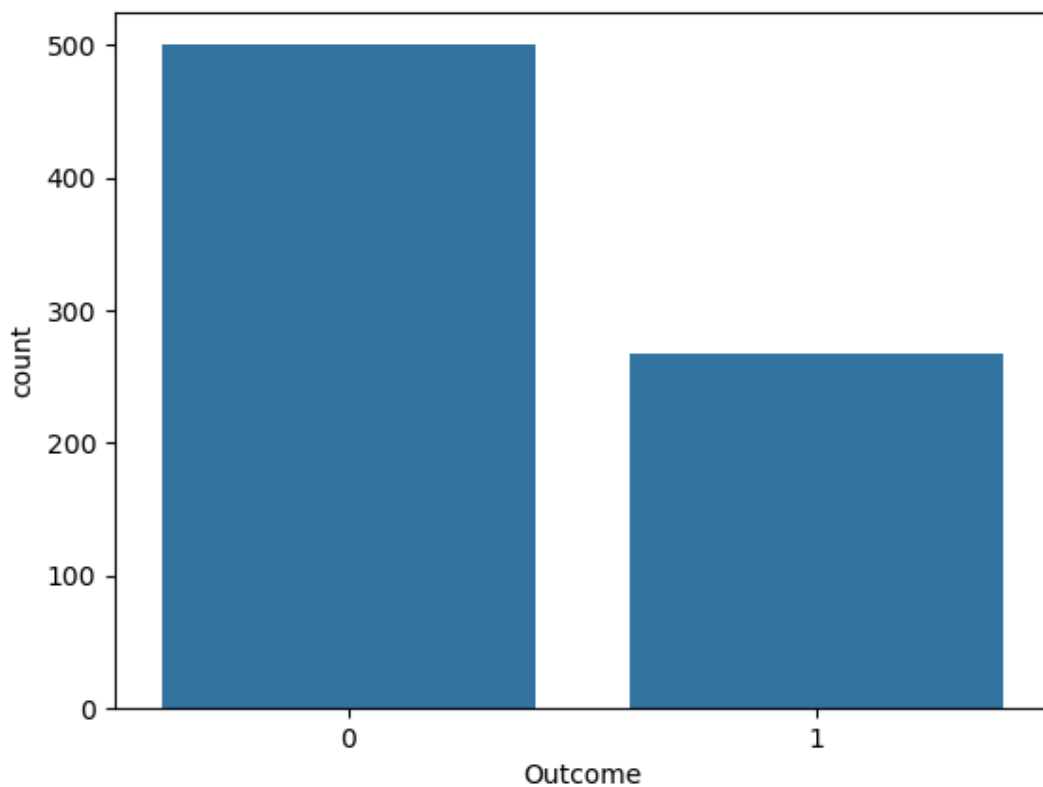
df = pd.read_csv("datasets/diabetes.csv")

#####
# Target'in Analizi
#####

df["Outcome"].value_counts()

sns.countplot(x="Outcome", data=df)
plt.show()

100 * df["Outcome"].value_counts() / len(df)
```



```
#####
# Feature'ların Analizi
```

```
#####
```

```
df.head()
```

```
df["BloodPressure"].hist(bins=20)
plt.xlabel("BloodPressure")
plt.show()
```

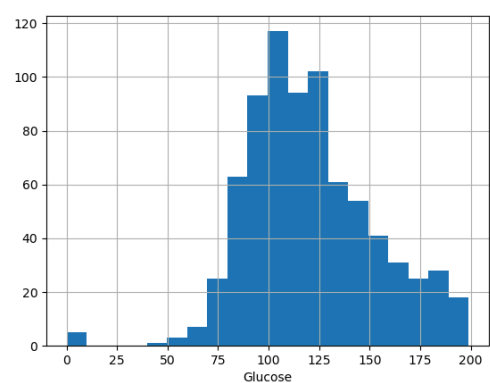
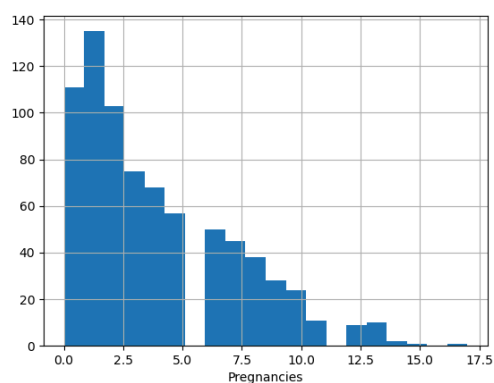
```
def plot_numerical_col(dataframe, numerical_col):
    dataframe[numerical_col].hist(bins=20)
    plt.xlabel(numerical_col)
    plt.show(block=True)
```

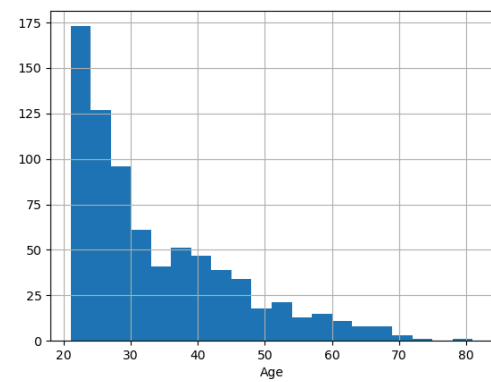
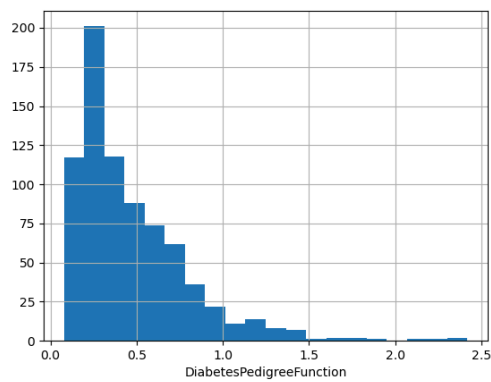
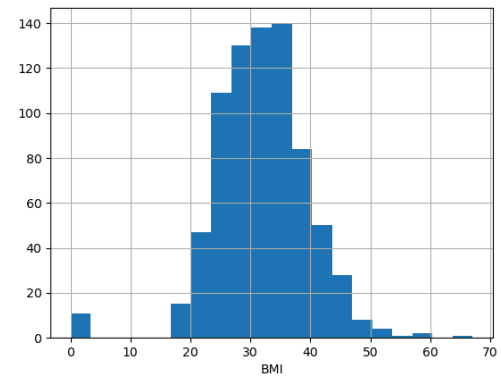
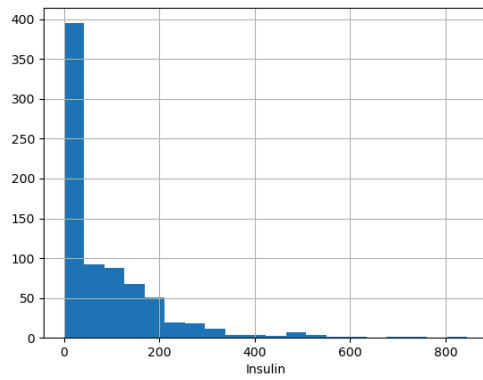
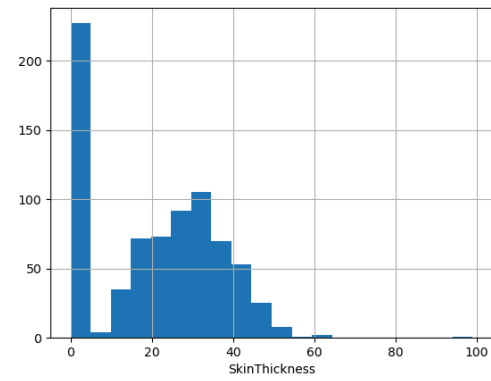
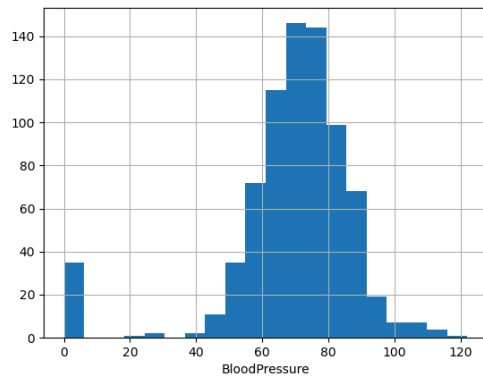
```
for col in df.columns:
    plot_numerical_col(df, col)
```

```
cols = [col for col in df.columns if "Outcome" not in col]
```

```
# for col in cols:
#     plot_numerical_col(df, col)
```

```
df.describe().T
```





```
#####
# Target vs Features
#####
```

```
df.groupby("Outcome").agg({"Pregnancies": "mean"})

def target_summary_with_num(dataframe, target, numerical_col):
    print(dataframe.groupby(target).agg({numerical_col: "mean"}), end="\n\n")

for col in cols:
    target_summary_with_num(df, "Outcome", col)
```

## Veri Ön İşleme ( Data Pre-Processing )

```
#####
# Data Preprocessing (Veri Ön İşleme)
#####
df.shape
df.head()

df.isnull().sum()

df.describe().T

for col in cols:
    print(col, check_outlier(df, col))

replace_with_thresholds(df, "Insulin")

for col in cols:
    df[col] = RobustScaler().fit_transform(df[[col]])

df.head()
```

**RobustScaler**, adından da anlaşılacağı gibi **aykırı değerlere karşı daha dayanıklıdır**. Medyan ve IQR, aykırı değerlerden ortalama ve standart sapmaya göre daha az etkilenir. Bu nedenle, verilerinde çok sayıda aykırı değer olduğunu düşündüğünde tercih edilebilir.

# Modelleme

```
#####  
# Model & Prediction  
#####  
  
y = df["Outcome"]  
  
X = df.drop(["Outcome"], axis=1)  
  
log_model = LogisticRegression().fit(X, y)  
  
log_model.intercept_  
log_model.coef_  
  
y_pred = log_model.predict(X)  
  
y_pred[0:10]  
  
y[0:10]
```

## Model Başarı Değerlendirme —> Train setinden test prediction yapıldı.

```
#####  
# Model Evaluation  
#####  
  
def plot_confusion_matrix(y, y_pred):  
    acc = round(accuracy_score(y, y_pred), 2)  
    cm = confusion_matrix(y, y_pred)  
    sns.heatmap(cm, annot=True, fmt=".0f")  
    plt.xlabel('y_pred')  
    plt.ylabel('y')  
    plt.title('Accuracy Score: {0}'.format(acc), size=10)  
    plt.show()
```

```
plot_confusion_matrix(y, y_pred)
```

```
print(classification_report(y, y_pred)) # scikit learn library'den
```

```
# Accuracy: 0.78
```

```
# Precision: 0.74
```

```
# Recall: 0.58
```

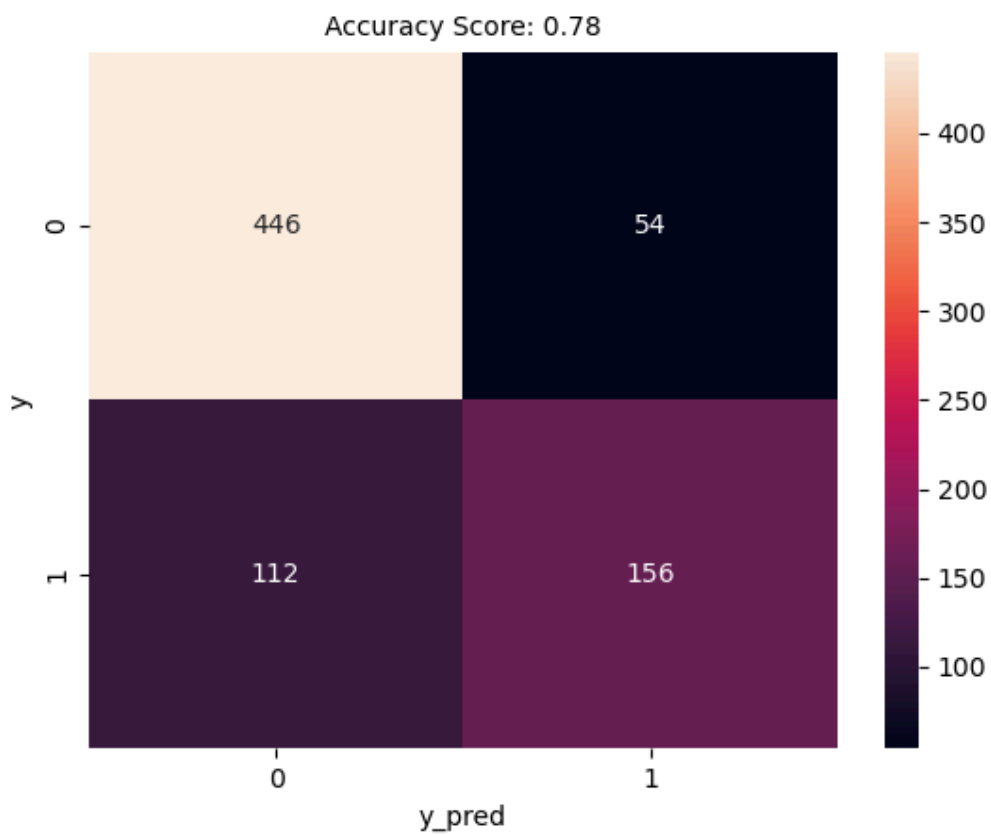
```
# F1-score: 0.65
```

```
# ROC AUC
```

```
y_prob = log_model.predict_proba(X)[:, 1]
```

```
roc_auc_score(y, y_prob)
```

```
# 0.83939
```





- **Accuracy:** Doğru sınıflandırma oranıdır:  $(TP+TN) / (TP+TN+FP+FN)$
- **Precision:** Pozitif sınıf (1) tahminlerinin başarı oranıdır:  $TP / (TP+FP)$ 
  - Hassasiyet /Precision modelin doğru tahminlediği olumlu sonuçların oranını, kesinlik / accuracy ise tüm tahminlerin doğruluğunu ifade eder.
- **Recall:** Pozitif sınıfın (1) doğru tahmin edilme oranıdır:  $TP / (TP + FN)$
- **F1 SCORE:**  $2 * (Precision * Recall) / (Precision + Recall)$

		Tahmin Edilen Sınıf	
		Sınıf = 1	Sınıf = 0
Gerçek Sınıf	Sınıf = 1	True Pozitif (TP)	False Negatif (FN)
	Sınıf = 0	False Pozitif (FP)	True Negatif (TN)

	precision	recall	f1-score	support
0	0.80	0.89	0.84	500
1	0.74	0.58	0.65	268
accuracy			0.78	768
macro avg	0.77	0.74	0.75	768
weighted avg	0.78	0.78	0.78	768

## Model Doğrulama ( Model Validation )

### ROC Eğrisi (Receiver Operating Characteristic Curve) Nedir?

ROC eğrisi, bir ikili sınıflandırma modelinin farklı sınıflandırma eşiklerinde (threshold) performansını görselleştirmek için kullanılan bir grafikdir. Modelinizin pozitif sınıfı (bu durumda müşteri terk etme, yani **CHURN=1**) ne kadar iyi ayırdığını gösterir.

```
#####
# Model Validation: Holdout
#####

X_train, X_test, y_train, y_test = train_test_split(X,
                                                    y,
                                                    test_size=0.20, random_state=17)

log_model = LogisticRegression().fit(X_train, y_train)

y_pred = log_model.predict(X_test)
y_prob = log_model.predict_proba(X_test)[:, 1]

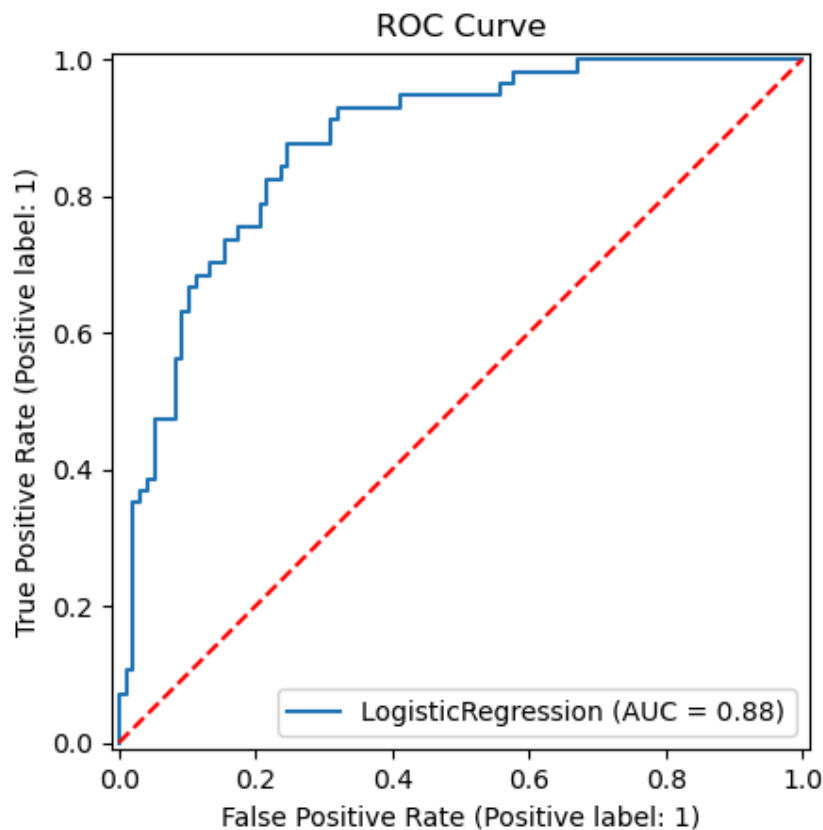
print(classification_report(y_test, y_pred))

# Accuracy: 0.78
# Precision: 0.74
# Recall: 0.58
# F1-score: 0.65

# Accuracy: 0.77
# Precision: 0.79
# Recall: 0.53
# F1-score: 0.63

RocCurveDisplay(log_model, X_test, y_test)
RocCurveDisplay.from_estimator(estimator=log_model, X=X_test, y=y_test)
plt.title('ROC Curve')
plt.plot([0, 1], [0, 1], 'r--')
plt.show()

# AUC → 0.8755652016639537
roc_auc_score(y_test, y_prob)
```



## 10 Katlı Çapraz Doğrulama ( 10-Fold Cross Validation )—> Hangi Train /test

```
#####
# Model Validation: 10-Fold Cross Validation
#####

y = df["Outcome"]
X = df.drop(["Outcome"], axis=1)

log_model = LogisticRegression().fit(X, y)

cv_results = cross_validate(log_model,
                             X, y,
                             cv=5,
                             scoring=["accuracy", "precision", "recall", "f1", "roc_auc"])
```

```
# Accuracy: 0.78
# Precision: 0.74
# Recall: 0.58
# F1-score: 0.65
```

```
# Accuracy: 0.77
# Precision: 0.79
# Recall: 0.53
# F1-score: 0.63
```

```
cv_results['test_accuracy'].mean()
# Accuracy: 0.7721
```

```
cv_results['test_precision'].mean()
# Precision: 0.7192
```

```
cv_results['test_recall'].mean()
# Recall: 0.5747
```

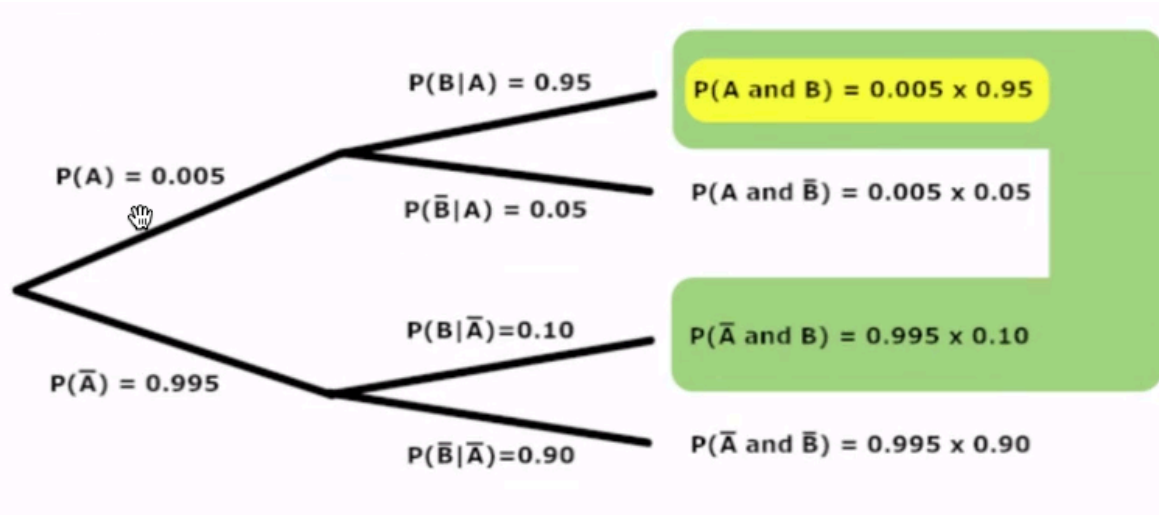
```
cv_results['test_f1'].mean()
# F1-score: 0.6371
```

```
cv_results['test_roc_auc'].mean()
# AUC: 0.8327
```

```
#####
# Prediction for A New Observation
#####
```

```
X.columns
```

```
random_user = X.sample(1, random_state=45)
log_model.predict(random_user)
```



### 📊 Confusion Matrix (Karmaşıklık Matrisi)

Aşağıdaki matris, tahmin ve gerçek değerlerin birleşiminden oluşur:

	Gerçek 1	Gerçek 0
Tahmin 1	TP	FP
Tahmin 0	FN	TN

Yani:

- Satırlar: Modelin tahmini
- Sütunlar: Gerçek sınıf

### 📈 Metrikler – Basit Anlatım ve Formüllerle

#### ✅ Accuracy (Doğruluk Oranı)

Modelin doğru tahmin ettiği tüm örneklerin oranı

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + FN + TN}$$

✓ Yani:

Kaç tane doğru bildi? (TP + TN)

Bölü: Tüm tahmin sayısı

