In [42]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
import statsmodels.formula.api as smf
import seaborn as sns
from sklearn.preprocessing import scale
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
from sklearn.metrics import roc_auc_score, roc_curve
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import GaussianNB
from sklearn import tree
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from xgboost import XGBClassifier
from lightgbm import LGBMClassifier
from catboost import CatBoostClassifier
from warnings import filterwarnings
import matplotlib.pyplot as plt

# Uyarıları göz ardı et
filterwarnings('ignore')
```

In [43]:
```
data = pd.read_csv("../input/ortopedik/ortopedik.csv")
df = data.copy()
df.head()
```

Out[43]:

	pelvic_incidence	pelvic_tilt numeric	lumbar_lordosis_angle	sacral_slope	pelvic_radius	degree_spondylolisthesis	class
0	63.027817	22.552586	39.609117	40.475232	98.672917	-0.254400	Abnormal
1	39.056951	10.060991	25.015378	28.995960	114.405425	4.564259	Abnormal
2	68.832021	22.218482	50.092194	46.613539	105.985135	-3.530317	Abnormal
3	69.297008	24.652878	44.311238	44.644130	101.868495	11.211523	Abnormal
4	49.712859	9.652075	28.317406	40.060784	108.168725	7.918501	Abnormal

Keşifsel Veri Analizi (EDA)

Bu veri seti, omurganın ortopedik sağlık durumu ile ilgili sınıflandırma yapmak için kullanılan bir veri kümesidir. Özellikle, hastaların omurga rahatsızlıklarını sınıflandırmak amacıyla kullanılır. Omurganın farklı özellikleri ölçülerek anormal ya da normal bir durumu teşhis etmeyi hedefler.

In [44]:
```
df.info()
```

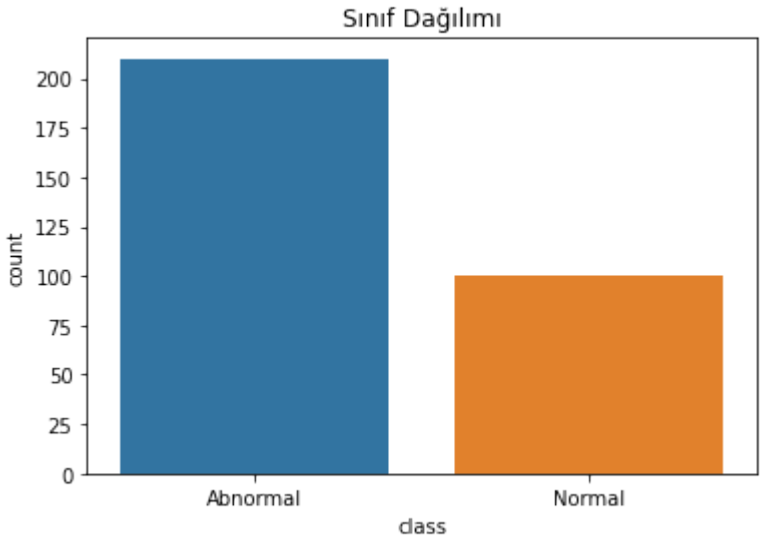
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 310 entries, 0 to 309
Data columns (total 7 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   pelvic_incidence          310 non-null    float64
 1   pelvic_tilt numeric       310 non-null    float64
 2   lumbar_lordosis_angle     310 non-null    float64
 3   sacral_slope              310 non-null    float64
 4   pelvic_radius             310 non-null    float64
 5   degree_spondylolisthesis  310 non-null    float64
 6   class                     310 non-null    object 
dtypes: float64(6), object(1)
memory usage: 17.1+ KB
```

Veri Seti Özeti: Kayıt Sayısı: 310 Özellik Sayısı: 7

Özelliklerin Tanımları

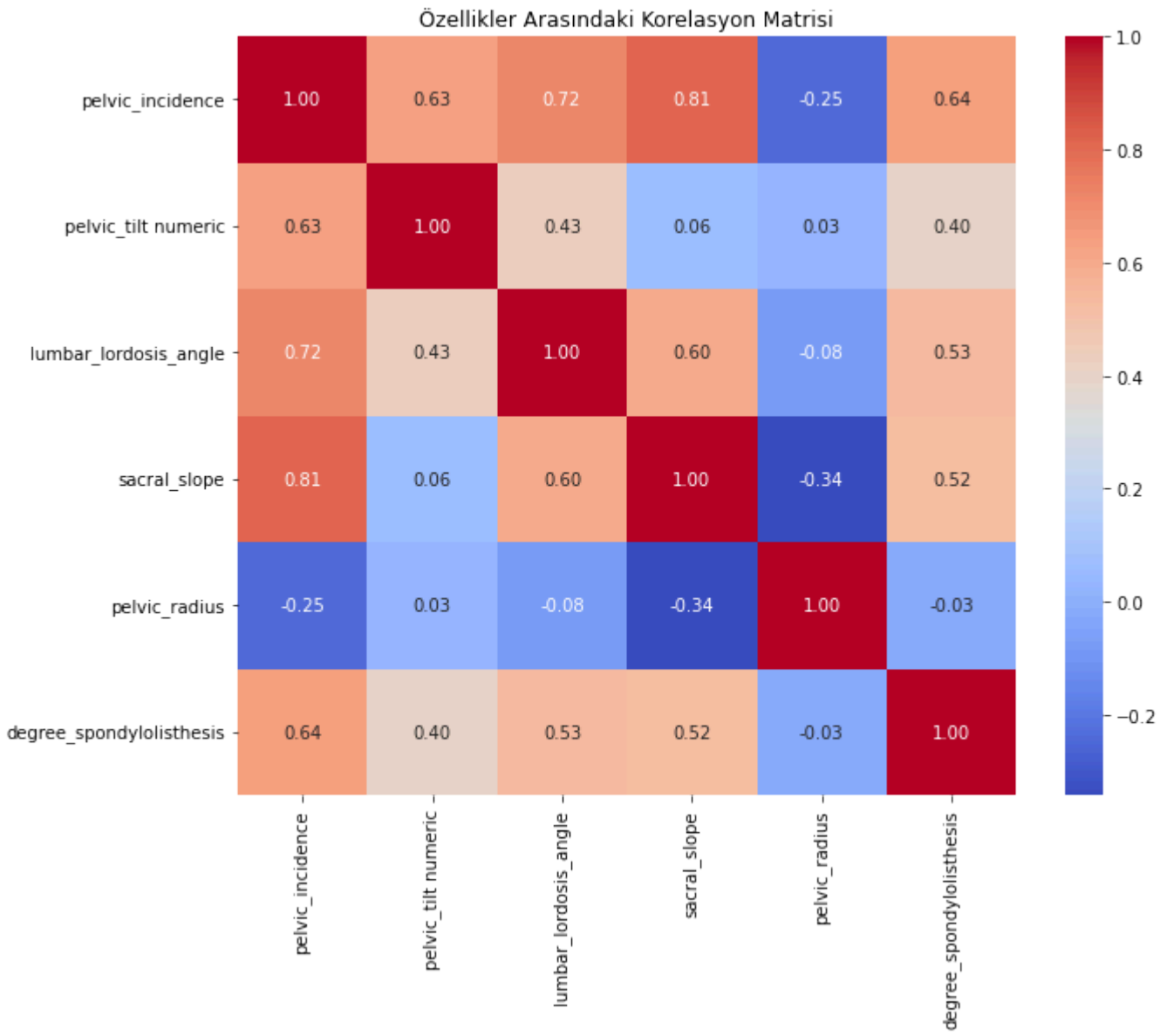
- **pelvic_incidence (PI):** Pelvis insidansı. Pelvis'in eğimini ölçen bir açı.
- **pelvic_tilt numeric (PT):** Pelvis eğilimi. Pelvis'in dikey eksene göre açısını ölçer.
- **lumbar_lordosis_angle (LLA):** Lomber lordoz açısı. Omurganın alt kısmındaki doğal iç bükey eğrinin açısını ölçer.
- **sacral_slope (SS):** Sakral eğim. Sakrum kemiğinin yatay eksene göre açısını ölçer.
- **pelvic_radius (PR):** Pelvis yarıçapı. Pelvis'in anatomik büyüklüğünü temsil eder.
- **degree_spondylolisthesis (DS):** Spondilolistezis derecesi. Omurların ileri-geri kaymasını ölçer.
- **class:** Hedef değişken. İki sınıf içerir: Abnormal: Anormal durum. Normal: Normal durum.

In [45]:
```
sns.countplot(x='class', data=df)
plt.title('Sınıf Dağılımı')
plt.show()
```



Bağımlı değişkenimizin (Class) sınıf dağılımları: "Abnormal" sınıfında 210 kayıt varken, "Normal" sınıfında sadece 100 kayıt bulunuyor.

In [46]:
```
# Sayısal sütunlar arasındaki korelasyonları hesaplayalım
corr_matrix = df.corr()

# Korelasyonları görselleştirme
plt.figure(figsize=(10, 8))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Özellikler Arasındaki Korelasyon Matrisi')
plt.show()
```



Korelasyon Analizi

- pelvic_incidence ve sacral_slope arasında yüksek korelasyon mevcut.
- lumbar_lordosis_angle ve sacral_slope arasında da anlamlı bir pozitif korelasyon var.
- pelvic_incidence, lumbar_lordosis_angle ve sacral_slope genel olarak diğer özelliklerle pozitif korelasyona sahip.
- pelvic_radius ile diğer özellikler arasında zayıf korelasyon var.

In [47]:
```
summary_stats = df.groupby('class').describe().T
summary_stats
```

Out[47]:

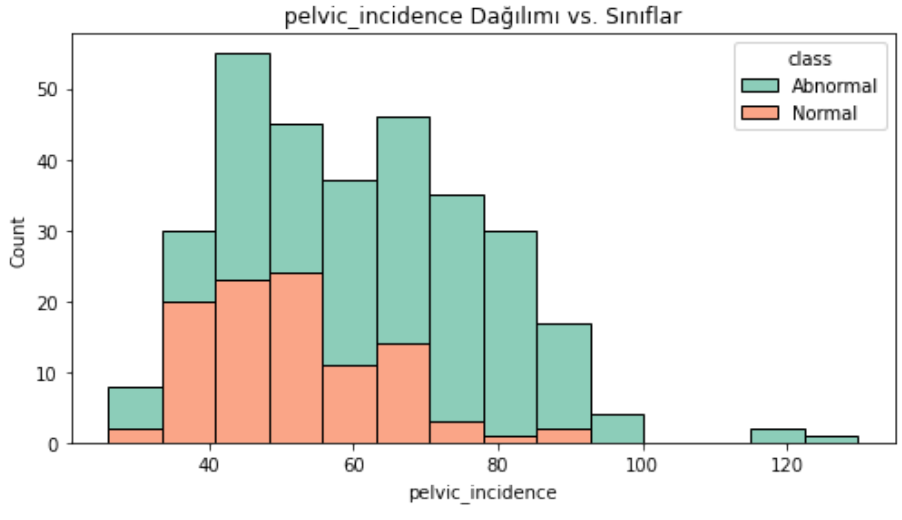
	class	Abnormal	Normal
pelvic_incidence	**count**	210.000000	100.000000
	mean	64.692562	51.685244
	std	17.662129	12.368161
	min	26.147921	30.741938
	25%	50.102507	42.817849
	50%	65.274888	50.123115
	75%	77.593672	61.470097
	max	129.834041	89.834676
pelvic_tilt numeric	**count**	210.000000	100.000000
	mean	19.791111	12.821414
	std	10.515871	6.778503
	min	-6.554948	-5.845994
	25%	13.048130	8.799951
	50%	18.798899	13.482435
	75%	24.815515	16.785953
	max	49.431864	29.894119
lumbar_lordosis_angle	**count**	210.000000	100.000000
	mean	55.925370	43.542605
	std	19.669471	12.361388
	min	14.000000	19.071075
	25%	41.116964	35.000000
	50%	56.150000	42.638923
	75%	68.102805	51.602346
	max	125.742385	90.563461
sacral_slope	**count**	210.000000	100.000000
	mean	44.901450	38.863830
	std	14.515560	9.624004
	min	13.366931	17.386972
	25%	34.380345	32.340487
	50%	44.639597	37.059694
	75%	55.146868	44.608788
	max	121.429566	67.195460
pelvic_radius	**count**	210.000000	100.000000
	mean	115.077713	123.890834
	std	14.090605	9.014246
	min	70.082575	100.501192
	25%	107.309280	118.182659
	50%	115.650323	123.874328
	75%	123.133365	129.040401
	max	163.071041	147.894637
degree_spondylolisthesis	**count**	210.000000	100.000000
	mean	37.777705	2.186572
	std	40.696741	6.307483
	min	-10.675871	-11.058179
	25%	7.263227	-1.511360
	50%	31.946516	1.152710
	75%	55.371614	4.968807
	max	418.543082	31.172767

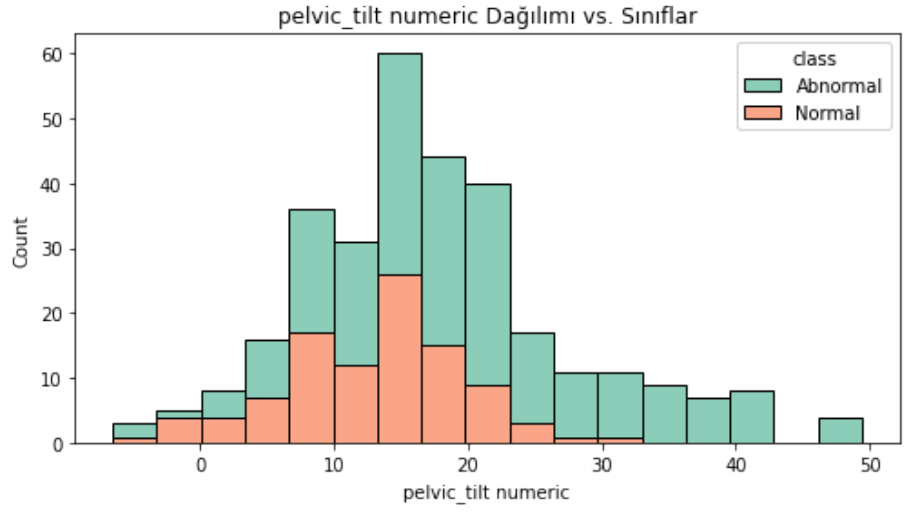
Özellik	Ölçüm	Abnormal	Normal
pelvic_incidence	Ortalama	64.69	51.69
	Standart Sapma	17.66	12.37
	Minimum	26.15	30.74
	Maksimum	129.83	89.83
pelvic_tilt numeric	Ortalama	19.79	12.82
	Standart Sapma	10.52	6.78
lumbar_lordosis_angle	Ortalama	55.93	43.54
	Standart Sapma	19.67	12.36
sacral_slope	Ortalama	44.90	38.86
	Standart Sapma	14.52	9.62
pelvic_radius	Ortalama	115.08	123.89
	Standart Sapma	14.09	9.01
degree_spondylolisthesis	Ortalama	37.78	2.19
	Standart Sapma	40.70	6.31

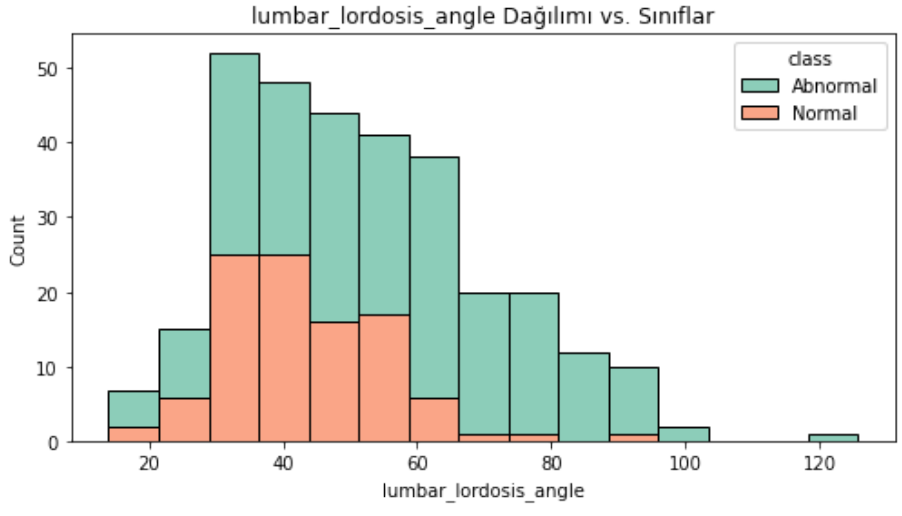
Yukarıda tabloda bağımsız değişkenlerin bağımlı değişkene göre betimsel istatistikleri verilmiştir.

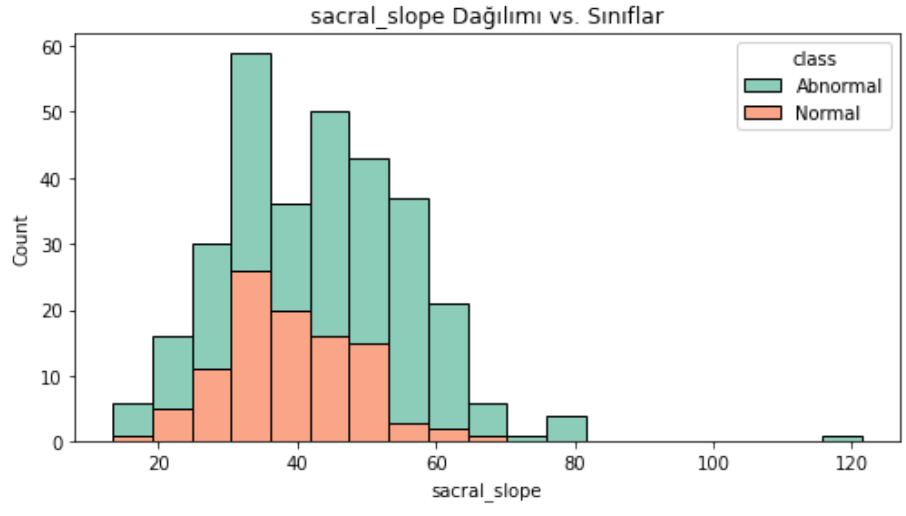
Veri Görselleştirme

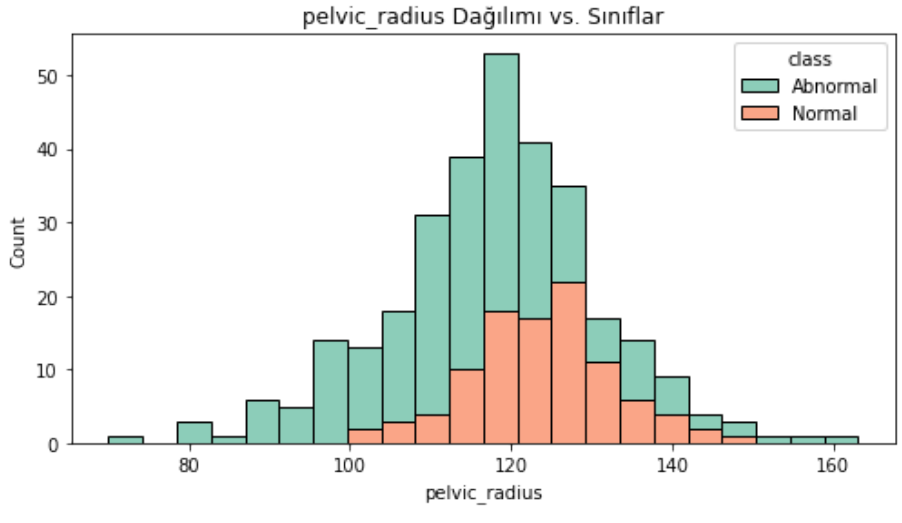
In [48]:
```
# Özellik dağılımı vs. Sınıflar
fig, axes = plt.subplots(nrows=3, ncols=2, figsize=(14, 12))
features = ['pelvic_incidence', 'pelvic_tilt numeric', 'lumbar_lordosis_angle',
            'sacral_slope', 'pelvic_radius', 'degree_spondylolisthesis']
axes = axes.flatten()

for i, feature in enumerate(features):
    sns.histplot(data=df, x=feature, hue='class', multiple='stack', palette='Set2', ax=axes[i])
    axes[i].set_title(f'{feature} Dağılımı vs. Sınıflar')

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

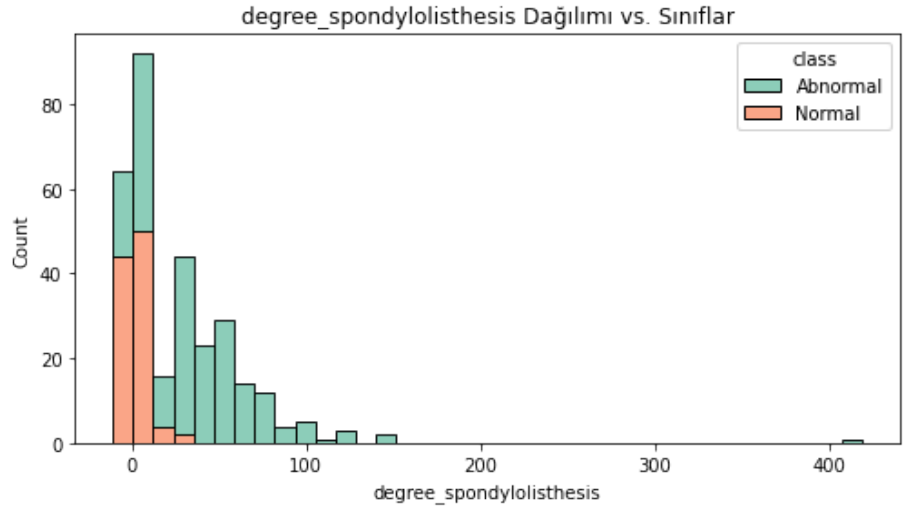




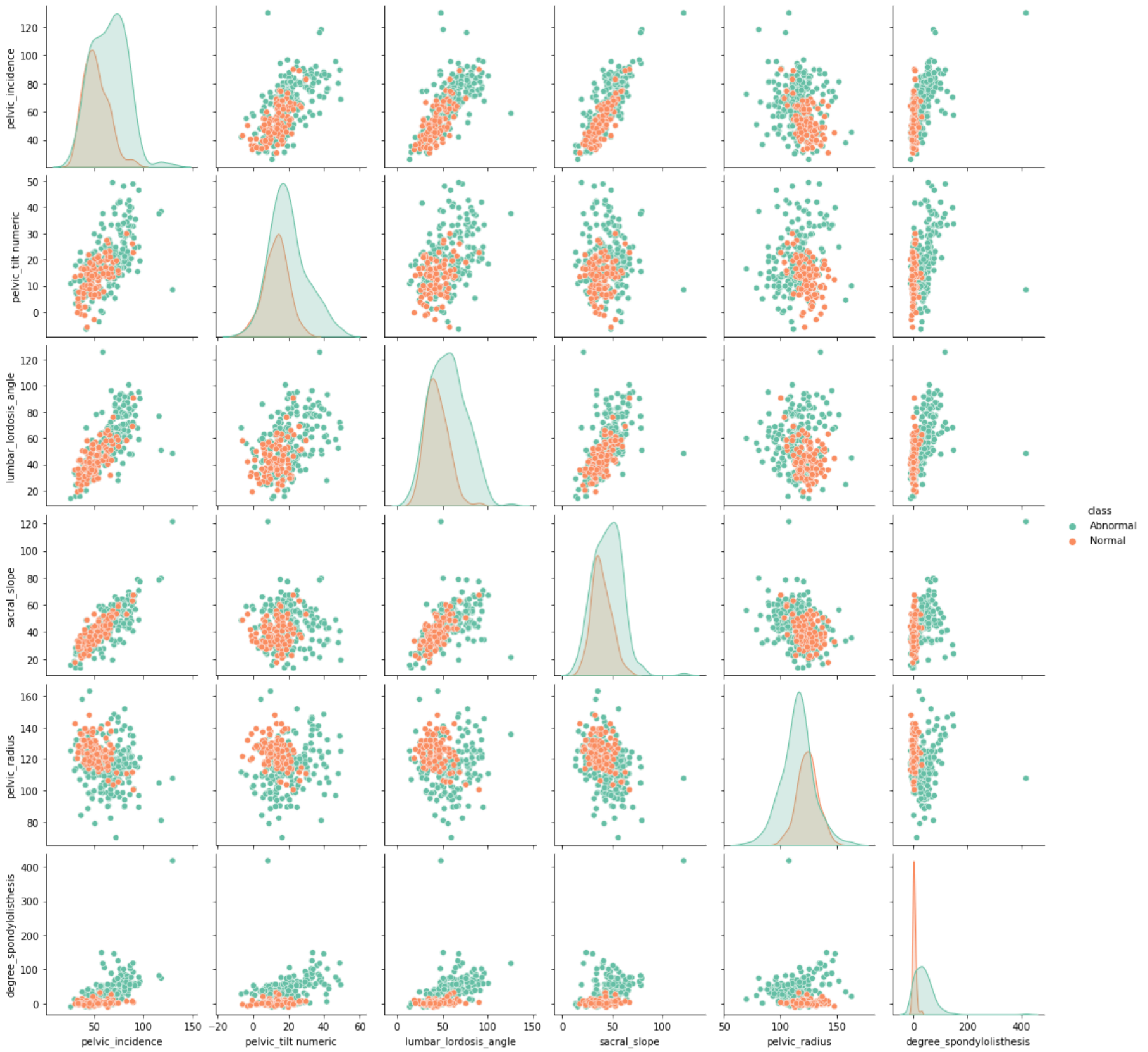




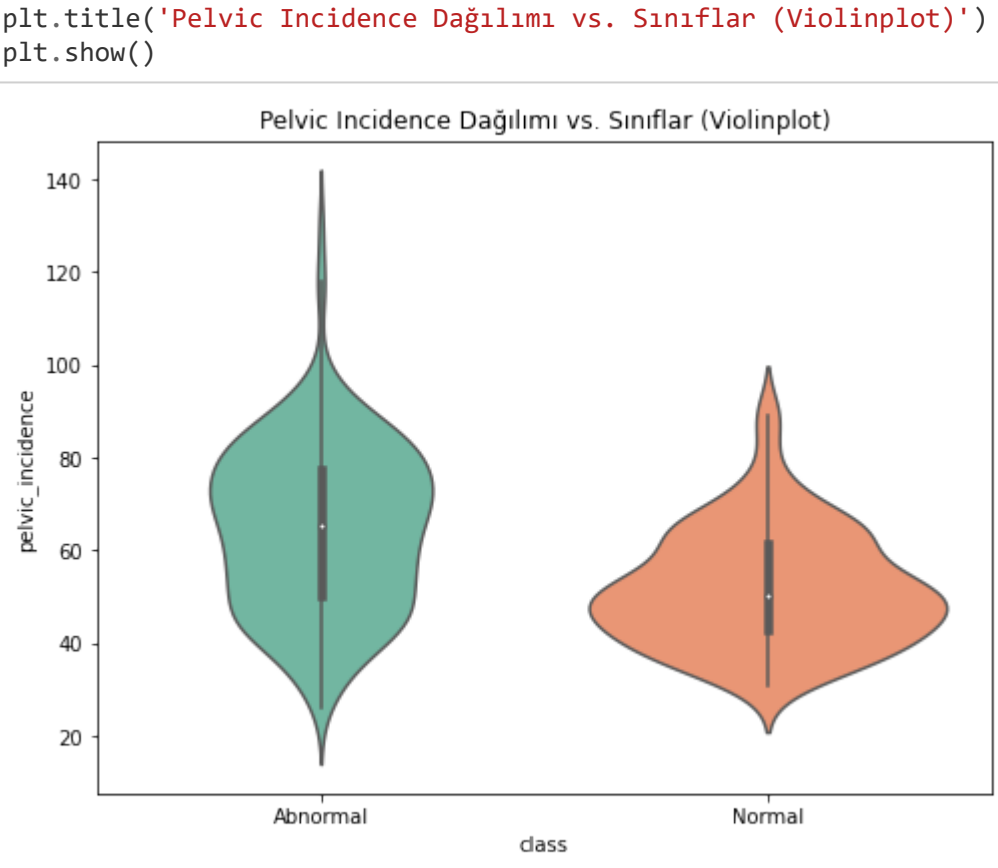




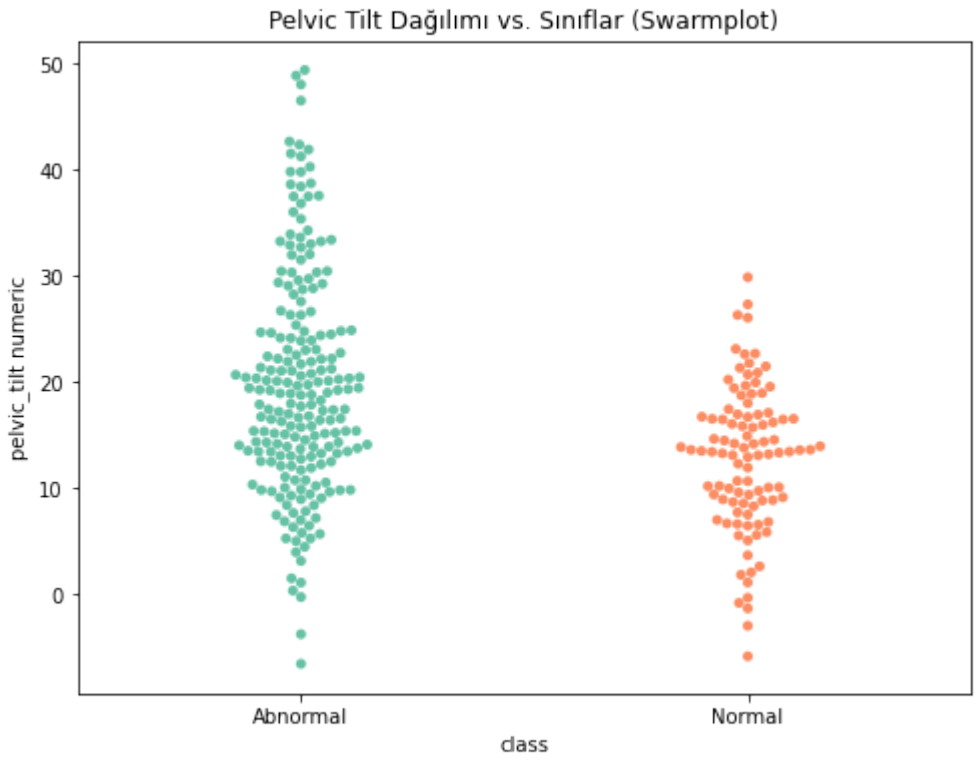
In [49]:
```
# Pairplot (Özellikler Arası İlişkiler)
sns.pairplot(df, hue='class', palette='Set2')
plt.show()
```



In [50]:
```
# Violinplot (Pelvic Incidence vs. Sınıflar)
plt.figure(figsize=(8, 6))
sns.violinplot(data=df, x='class', y='pelvic_incidence', palette='Set2')
plt.title('Pelvic Incidence Dağılımı vs. Sınıflar (Violinplot)')
plt.show()
```



In [51]:
```
# Swarmplot (Pelvic Tilt vs. Sınıflar)
plt.figure(figsize=(8, 6))
sns.swarmplot(data=df, x='class', y='pelvic_tilt numeric', palette='Set2')
plt.title('Pelvic Tilt Dağılımı vs. Sınıflar (Swarmplot)')
plt.show()
```



Veri Ön İşleme

In [52]:
```
df['class'] = df['class'].apply(lambda x: 1 if x == 'Abnormal' else 0)
```

- **Class sütununu sayısala çevirdik. Binary Sınıflandırma yapacağımız için 1 ve 0 olarak ayarladık. (1: Abnormal, 0: Normal)**

In [53]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 310 entries, 0 to 309
Data columns (total 7 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   pelvic_incidence          310 non-null    float64
 1   pelvic_tilt numeric       310 non-null    float64
 2   lumbar_lordosis_angle     310 non-null    float64
 3   sacral_slope              310 non-null    float64
 4   pelvic_radius             310 non-null    float64
 5   degree_spondylolisthesis  310 non-null    float64
 6   class                     310 non-null    int64  
dtypes: float64(6), int64(1)
memory usage: 17.1 KB
```

- Yukarıda değişkenlerimizin Dtype değerlerinde bütün değerlerin sayısal olduğu görünmektedir. Şimdi eksik değerlerimiz varmı onu inceleyelim, daha sonra aykırı değer incelemesi yapalım.
- Eğer diğer değerlerimizde sayısal tipte olmasaydı, sayısala çevirmemiz gerekiyordu. Makine öğrenmesine verebilmemiz için bütün değerlerimiz sayısal olması gerekiyor genel olarak.

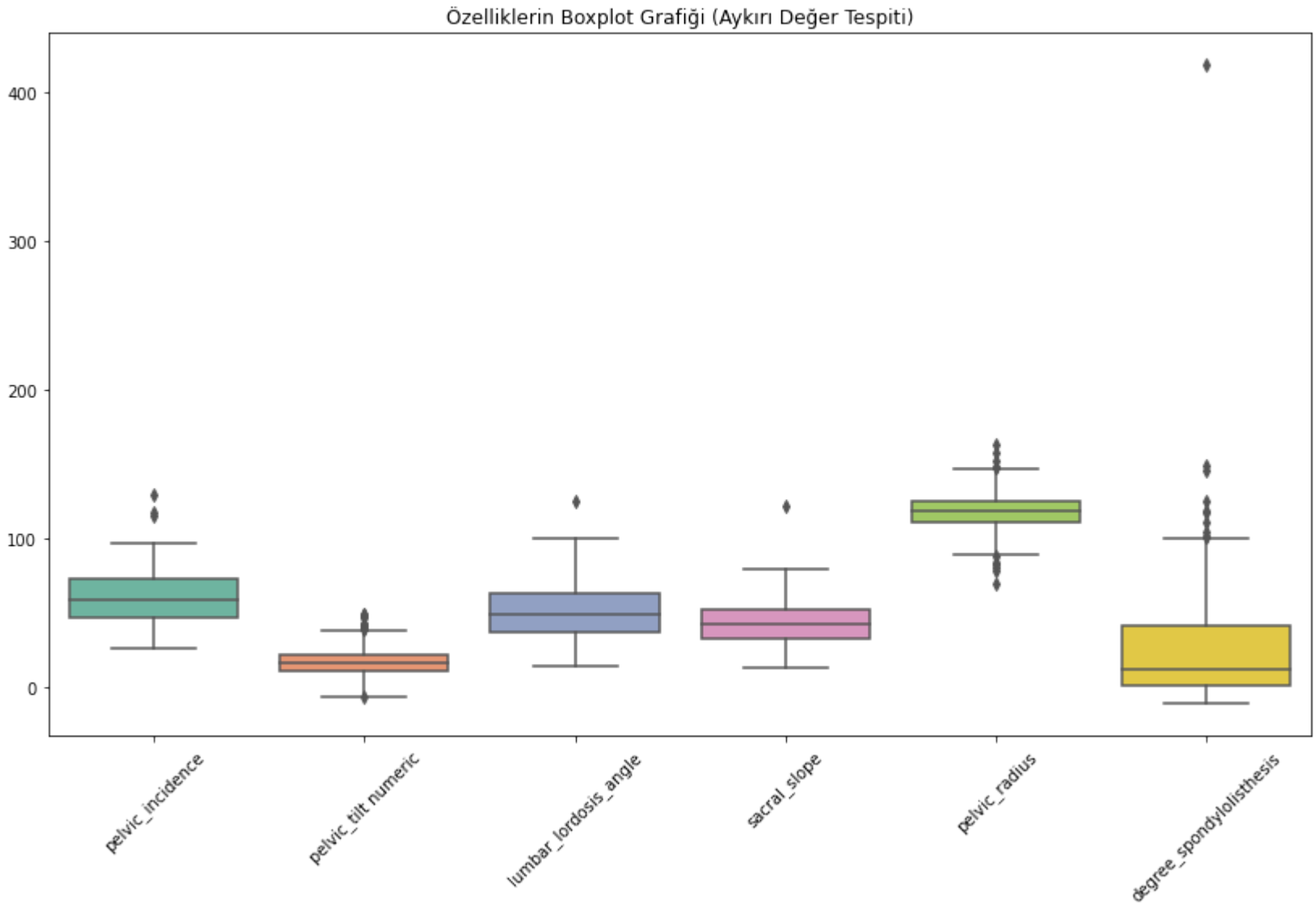
In [54]:
```
eksikdeğerler = df.isnull().sum()
print(eksikdeğerler)
```

```
pelvic_incidence          0
pelvic_tilt numeric       0
lumbar_lordosis_angle     0
sacral_slope              0
pelvic_radius             0
degree_spondylolisthesis  0
class                     0
dtype: int64
```

Yukarıdaki setimizi incelediğimizde eksik değer görünmemektedir.Eğer eksik değerler olsaydı, eksik değerden kurtulma yöntemleri:

- Eksik değer silinebilir,
- ortalama ile doldurulabilir,
- en çok tekrar eden değer ile doldurulabilir.

In [55]:
```
# Aykırı değerleri incelemek için boxplot
plt.figure(figsize=(14, 8))
sns.boxplot(data=df.drop(columns=['class']), palette='Set2')
plt.title('Özelliklerin Boxplot Grafiği (Aykırı Değer Tespiti)')
plt.xticks(rotation=45)
plt.show()
```

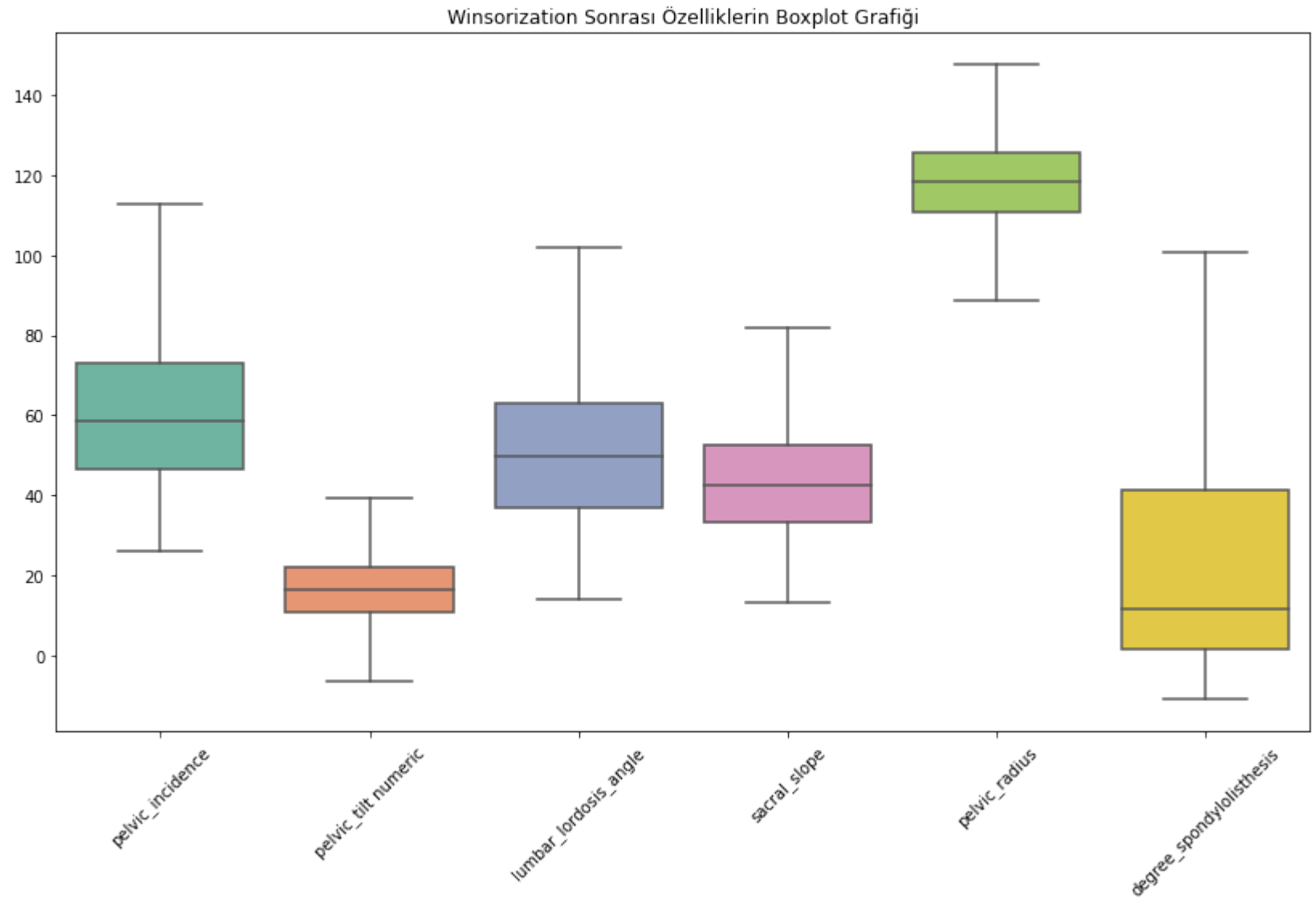


- Aykırı değerlerimizi Alt ve üst sınırlar (Çeyreklikler) belirlenerek, aykırı değerler bu sınırlara çekilecektir. Bu yönteme genellikle **Winsorization** denir. Baskılama yöntemi olarak da geçmektedir.

In [56]:
```
# Winsorization fonksiyonu
def winsorize_iqr(df, column):
    Q1 = df[column].quantile(0.25)
    Q3 = df[column].quantile(0.75)
    IQR = Q3 - Q1
    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR
    df[column] = df[column].apply(lambda x: max(min(x, upper_bound), lower_bound))

# Tüm sayısal sütunlar için Winsorization uygulama
for feature in features:
    winsorize_iqr(df, feature)
```

winsorize_iqr fonksiyonu ile baskılama işlemi yapılmaktadır. İki tane parametre almaktadır. İlk parametre veri seti, ikinci parametre kolonları (Özellikleri) almaktadır. Biz yukarı kısımda (Veri Görselleştirme) features değişkenine kolonları atamıştık. For döngüsü ile kolonları teker teker fonksiyona verdik. Böylece aykırı değerler alt limite ve üst limite eşitlenmiş oldu.

In [57]:
```
plt.figure(figsize=(14, 8))
sns.boxplot(data=df.drop(columns=['class']), palette='Set2')
plt.title('Winsorization Sonrası Özelliklerin Boxplot Grafiği')
plt.xticks(rotation=45)
plt.show()
```



- Yukarıda aykırı değerlerin kalmadığı görünmektedir. Görsel aralığı 0-400 iken, 0 ile 140'a düştüğü için görsel daha büyük görünmektedir.

Makine Öğrenmesi

In [58]:
```
y = df["class"]
X = df.drop(['class'], axis=1)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)
```

- Y (bağımlı değişkenimiz) ve x bağımsız değişkenlerimizi ayarladık. Y sadece class sutunu, X ise class hariç bütün sutunları almaktadır.
- X_train: X eğitim için kullanılacak bağımsız değişkenler, X_test: Test seti, Y_train: Eğitim için bağımlı değişkenler, Y_Train: Test için bağımlı değişken.
- **Test setimiz %30, eğitim setimiz %70 olarak ayarlandı. Veri setimizdeki kayıtların az olması sebebiyle bu şekilde uygun görüldü. Eğer veri setimizde milyonlarca veri olsaydı %99 eğitim, %1 test olarak da ayarlanabilirdi.**

Gaussian Naive Bayes modeli

- Gaussian Naive Bayes, sürekli özelliklerin normal dağılım varsayımı altında sınıflandırma yapmak için Bayes teoremini kullanan olasılıksal bir makine öğrenmesi modelidir.

In [59]:
```
nb = GaussianNB()
nb_model = nb.fit(X_train, y_train)
nb_model
```

Out[59]:
```
GaussianNB()
```

- **Naive Bayes Modeli Kuruldu.**

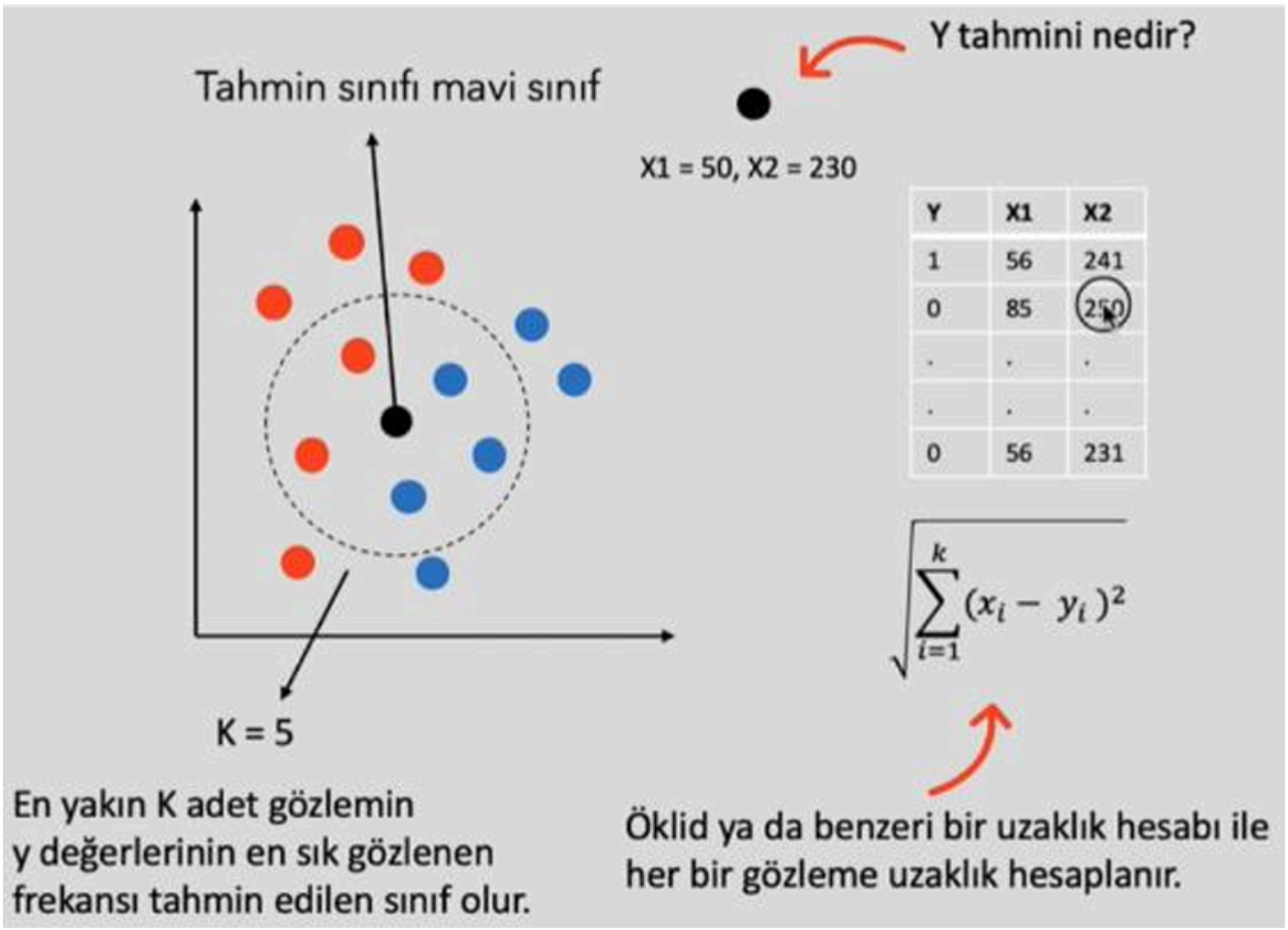
In [60]:
```
y_pred_nb = nb_model.predict(X_test)
print(f'Test Hatası: {accuracy_score(y_test, y_pred_nb)}')
```

```
Test Hatası: 0.7956989247311828
```

In [61]:
```
dogrulanmis_nb = cross_val_score(nb_model, X_test, y_test, cv = 10).mean()
print(f'Doğrulanmış Test Hatası: {dogrulanmis_nb}')
```

```
Doğrulanmış Test Hatası: 0.8066666666666666
```

K-En Yakın Komşu (KNN)



KNN yöntemi, veri kümesine bakarak verilere en yakın olan diğer verileri bulur ve bu verilerin sınıflarını göz önünde bulundurarak tahminler yapar. Örneğin, K değeri 3 olarak ayarlanmışsa, verilere en yakın 3 verinin sınıfları göz önünde bulundurularak tahmin yapılır. Bu yöntem, verilere en yakın olan verilerin sınıflarının çoğunlukta olduğu sınıfları tercih eder.

In [62]:
```
knn = KNeighborsClassifier()
knn_model = knn.fit(X_train, y_train)
knn_model
```

Out[62]:
```
KNeighborsClassifier()
```

In [63]:
```
y_pred_knn = knn_model.predict(X_test)
print(f'Test Hatası: {accuracy_score(y_test, y_pred_knn)}')
```

```
Test Hatası: 0.7849462365591398
```

In [64]:
```
dogrulanmis_knn = cross_val_score(knn_model, X_test, y_test, cv = 10).mean()
print(f'Doğrulanmış Test Hatası: {dogrulanmis_knn}')
```

```
Doğrulanmış Test Hatası: 0.8466666666666667
```

En iyi paremetreleri bulma (KNN)

In [65]:
```
knn_params = {"n_neighbors": np.arange(1,50)}
knn = KNeighborsClassifier()
knn_cv = GridSearchCV(knn, knn_params, cv=10)
knn_cv.fit(X_train, y_train)
```

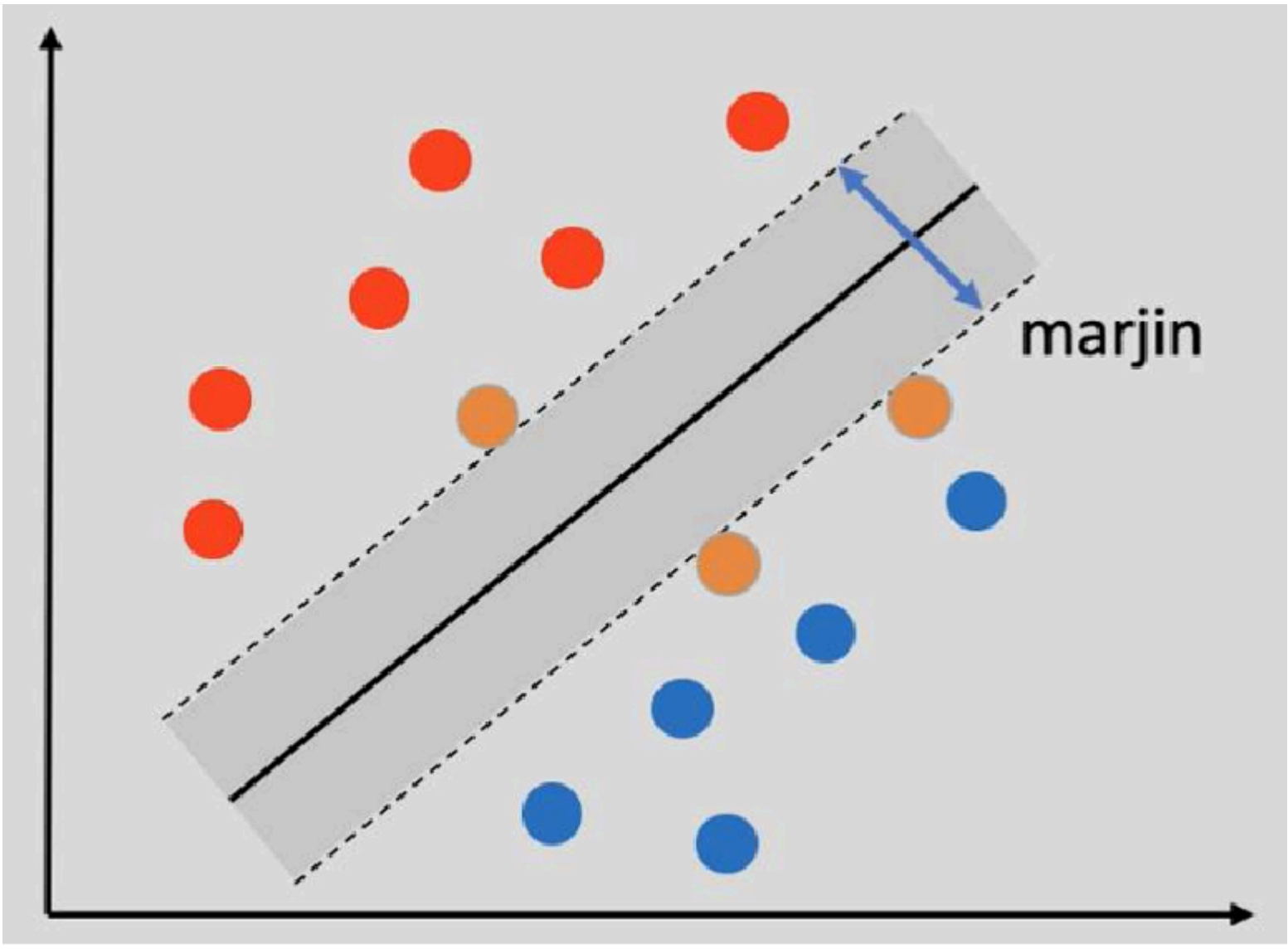
Out[65]:
```
GridSearchCV(cv=10, estimator=KNeighborsClassifier(),
             param_grid={'n_neighbors': array([ 1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17,
       18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
       35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49])})
```

- Komşu sayımız 1 den 50 ye kadar olan değerleri alsın.
- Komşu sayısı parametresini değiştirip deneyecek.

In [66]:
```
print("En iyi skor:" + str(knn_cv.best_score_))
print("En iyi parametre (Komşuluk değeri): " + str(knn_cv.best_params_))
```

```
En iyi skor:0.8523809523809524
En iyi parametre (Komşuluk değeri): {'n_neighbors': 13}
```

Destek Vektör Makineleri (SVM)



- **Destek Vektör Makineleri (SVM)**, iki sınıf arasındaki veri noktalarını ayırmak için en iyi ayrım çizgisini veya hiper düzlemi bulan ve bu ayrımı maksimize eden bir sınıflandırma algoritmasıdır.
- **Marjin:** Hiper düzlem ile en yakın veri noktası arasındaki alandır.
- **Hiperdüzlem:** İki doğru arasında nokta olmayacak şekilde aralarındaki mesafenin maksimum tutulması şekliyle sağlanır.

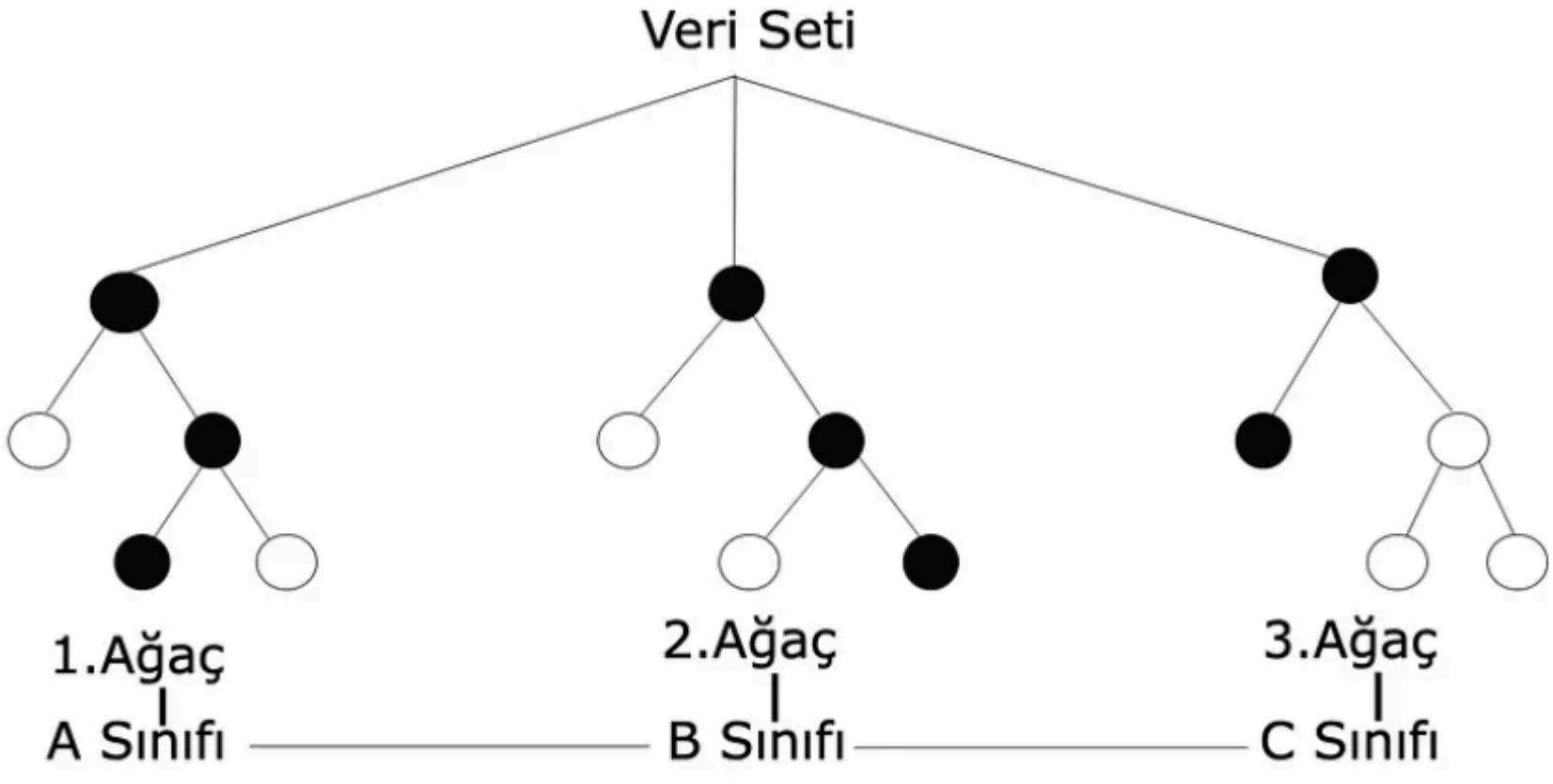
In [67]:
```
svm_model = SVC(kernel = "linear").fit(X_train, y_train)
y_pred_svm = svm_model.predict(X_test)
print(f'Test Hatası: {accuracy_score(y_test, y_pred_svm)}')
```

```
Test Hatası: 0.8924731182795699
```

In [68]:
```
dogrulanmis_svm = cross_val_score(svm_model, X_test, y_test, cv = 10).mean()
print(f'Doğrulanmış Test Hatası: {dogrulanmis_svm}')
```

```
Doğrulanmış Test Hatası: 0.8800000000000001
```

Random Forest Classifier



Random Forest Classifier, çok sayıda karar ağacı oluşturur ve her bir ağacın tahminlerini bir araya getirir. Bu sayede, model daha doğru tahminler yapar çünkü her bir ağaç farklı veri kümelerinden öğrenir ve farklı kurallar oluşturur. Bu, overfitting (aşırı uyum) sorununu azaltır ve modelin genelleştirme yeteneğini artırır.

In [69]:
```
rf_model = RandomForestClassifier().fit(X_train,y_train)
```

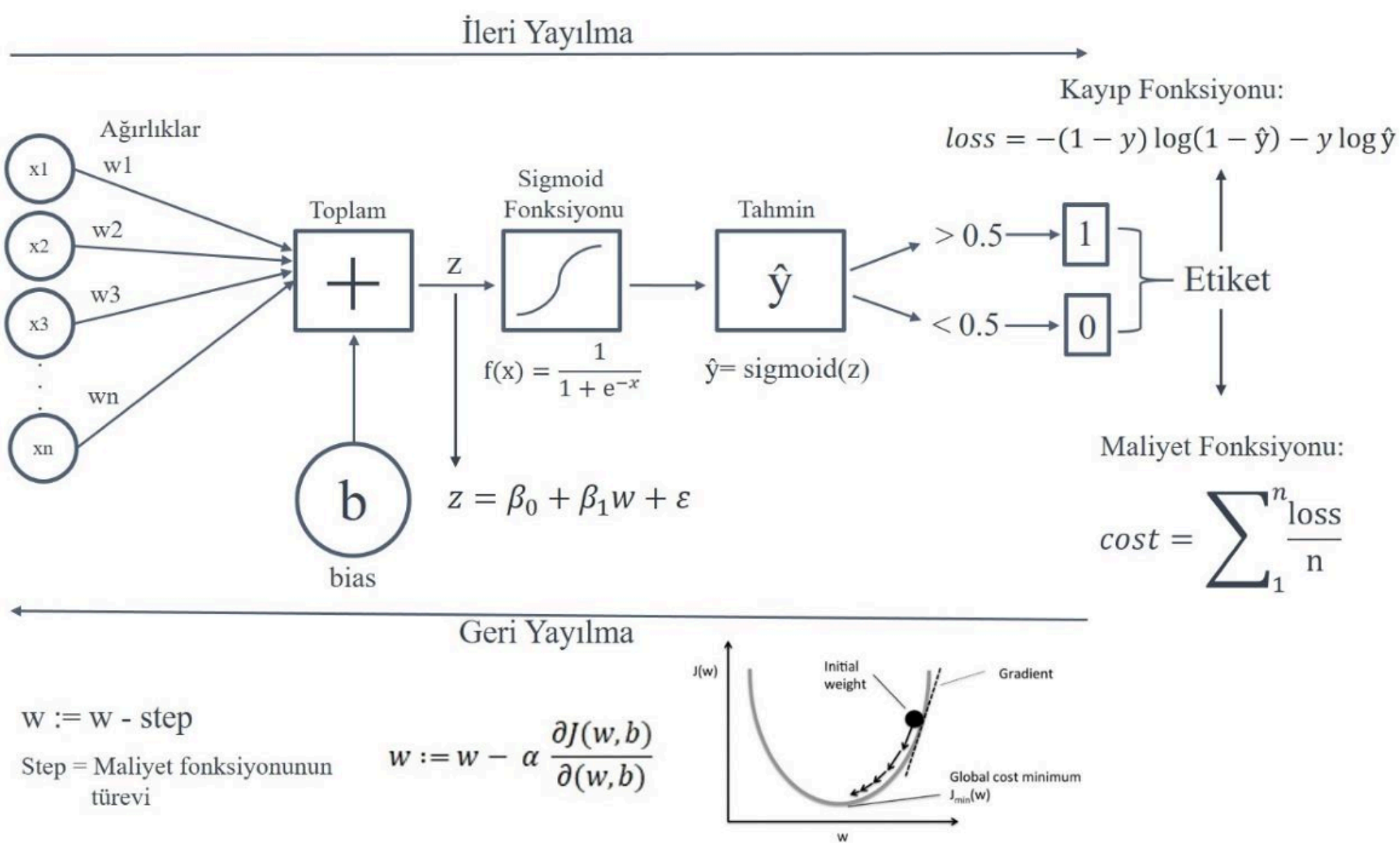
In [70]:
```
y_pred_rf=rf_model.predict(X_test)
 accuracy_score(y_test,y_pred_rf)
```

Out[70]: 0.8602150537634409

In [71]:
```
dogrulanmis_rf = cross_val_score(rf_model, X_test, y_test, cv = 10).mean()
print(f'Doğrulanmış Test Hatası: {dogrulanmis_rf}')
```

```
Doğrulanmış Test Hatası: 0.8466666666666667
```

Lojistik Regresyon



Lojistik Regresyon, bir veri kümesine bakarak verilerin iki farklı sınıfa ait olma olasılıklarını tahmin eder. Bu olasılıklar, 0 ile 1 arasında değerler alır ve 1, verinin sınıf A'ya ait olma olasılığının maksimum olduğunu gösterir. Lojistik Regresyon, verileri iki sınıf arasında en iyi şekilde böler ve bu sayede daha doğru tahminler yapar.

In [72]:
```
lojistik=LogisticRegression(solver="liblinear")
loj_model=lojistik.fit(X,y)
```

In [73]:
```
y_pred_loj=loj_model.predict(X_test)
accuracy_score(y_test,y_pred_loj)
```

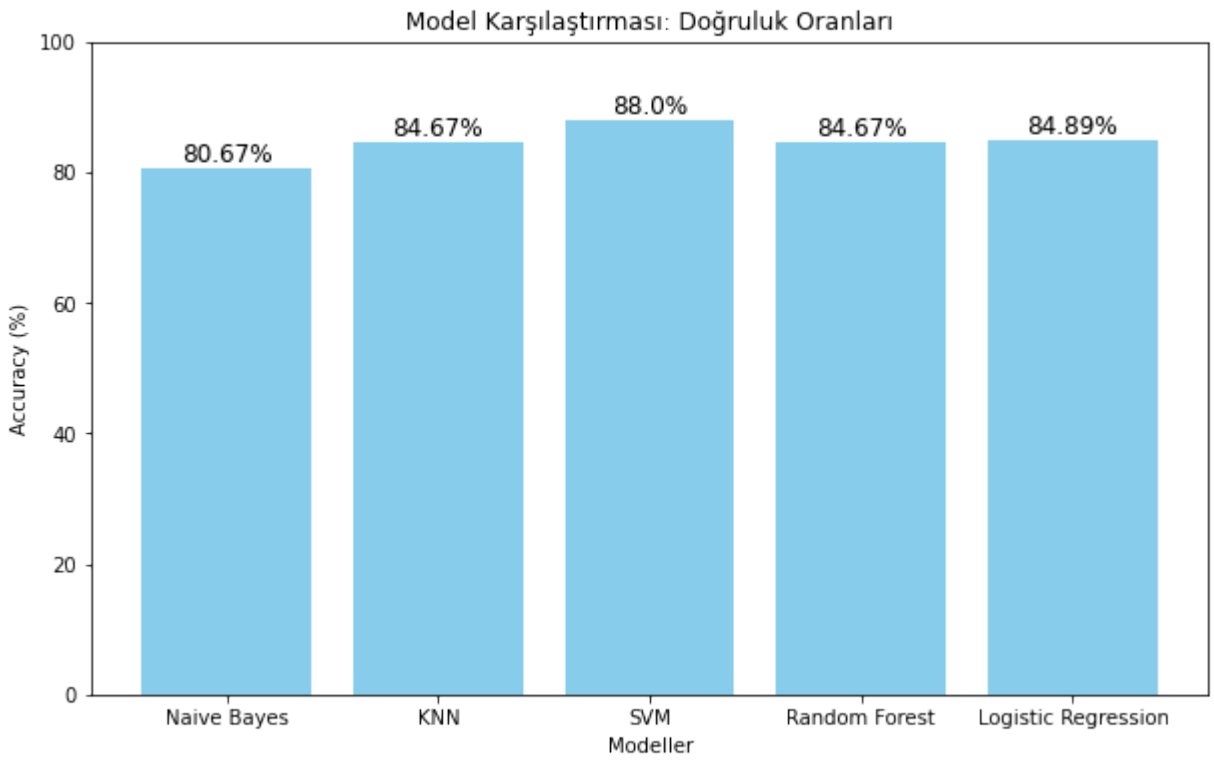
Out[73]: 0.8387096774193549

In [74]:
```
dogrulanmis_loj = cross_val_score(loj_model, X_test, y_test, cv = 10).mean()
print(f'Doğrulanmış Test Hatası: {dogrulanmis_loj}')
```

```
Doğrulanmış Test Hatası: 0.8488888888888889
```

Model Karşılaştırılması

In [75]:
```
modeller=['Naive Bayes', 'KNN', 'SVM', 'Random Forest', 'Logistic Regression']

dogrulanmis_sonuclar=[
 dogrulanmis_nb,
 dogrulanmis_knn,
 dogrulanmis_svm,
 dogrulanmis_rf,
 dogrulanmis_loj
]

# Yüzdeye dönüştürme
dogrulanmis_sonuclar_percent = [round(x * 100, 2) for x in dogrulanmis_sonuclar]

# Görselleştirme
plt.figure(figsize=(10, 6))
plt.bar(modeller, dogrulanmis_sonuclar_percent, color='skyblue')
plt.ylabel('Accuracy (%)')
plt.xlabel('Modeller')
plt.title('Model Karşılaştırması: Doğruluk Oranları')
plt.ylim(0, 100)

# Her sütunun üstüne doğruluk oranını yazma
for i, v in enumerate(dogrulanmis_sonuclar_percent):
    plt.text(i, v + 1, f'{v}%', ha='center', fontsize=12)

plt.show()
```



Modellerin toplu olarak doğruluk değerler getirdik. En yüksek değere sahip olan SVM.

Confusion Matrix

In [76]:
```
model_tahminleri=[
 y_pred_nb,
 y_pred_knn,
 y_pred_svm,
 y_pred_rf,
 y_pred_loj
]

for i in range(len(model_tahminleri)):
    print("{0} Modelimizin Confusion Matrix:\n {1}".format(modeller[i],confusion_matrix(y_test,model_tahminleri[i])))
```

```
Naive Bayes Modelimizin Confusion Matrix:
 [[20  4]
 [15 54]]
KNN Modelimizin Confusion Matrix:
 [[16  8]
 [12 57]]
SVM Modelimizin Confusion Matrix:
 [[20  4]
 [ 6 63]]
Random Forest Modelimizin Confusion Matrix:
 [[16  8]
 [ 5 64]]
Logistic Regression Modelimizin Confusion Matrix:
 [[18  6]
 [ 9 60]]
```

- Confusion Matrix değerlerimizi incelediğimizde Tip 2 hatasının en az olduğu model Random Forest Modelidir (RF). Doğruluk olarak SVM modeli daha başarılı çıksa da Confusion Matrix'de RF modeli daha başarılı olmuştur.
- **Tip1: 8 ve Tip2: 5**

F1Score,Precision,Recall ölçüm metrikleri

In [77]:
```
print(classification_report(y_test,model_tahminleri[2]))
```

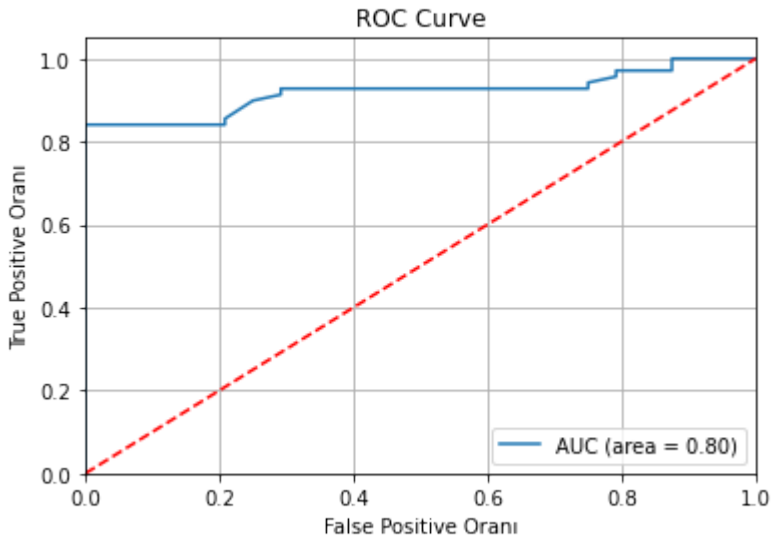
```
              precision    recall  f1-score   support

           0       0.77      0.83      0.80        24
           1       0.94      0.91      0.93        69

    accuracy                           0.89        93
   macro avg       0.85      0.87      0.86        93
weighted avg       0.90      0.89      0.89        93
```

Sınıf 0 (Normal)

- **Precision (Kesinlik): 0.77**
  - Sınıf 0 olarak tahmin edilen örneklerin %77'ü gerçekten doğru tahmin edilmiştir.
  - Yanlış pozitif sayısının fazla olduğu, yani bu sınıf için yanlış pozitif tahminler olduğu anlamına gelir.
- **Recall (Duyarlılık): 0.83**
  - Sınıf 0'a ait olan gerçek örneklerin sadece %83'i doğru tahmin edilmiştir.
  - Modelin, sınıf 0 örneklerini tanımakta zorlandığını gösterir (yanlış negatif sayısı yüksek).
- **F1-score: 0.80**
  - Precision ve Recall değerlerinin harmonik ortalamasıdır.
  - Düşük precision ve recall değerlerinin bir sonucu olarak 0.80 gibi bir skor elde edilmiştir.
- **Support: 24**
  - Test setindeki toplam sınıf 0 örnek sayısıdır.

Roc Eğrisi

In [79]:
```
# Test setinde ROC AUC puanını hesaplayalım
rf_roc_auc = roc_auc_score(y_test, rf_model.predict(X_test))

# Test setinde ROC eğrisini hesaplayalım
fpr, tpr, thresholds = roc_curve(y_test, rf_model.predict_proba(X_test)[:, 1], pos_label=1)

# ROC eğrisini çizelim
plt.figure()
plt.plot(fpr, tpr, label='AUC (area = %0.2f)' % rf_roc_auc)
plt.plot([0, 1], [0, 1], 'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Oranı')
plt.ylabel('True Positive Oranı')
plt.title('ROC Curve')
plt.legend(loc='lower right')
plt.grid(True)
plt.show()
```



- **Mavi çizgi modelin başarısını hesaplamak, değerledirmek için kullanacak olduğumuz eğri**
- **Kırmızı çizgi hiçbir modelleme yapmasaydık zaten elde edeceğimiz başarıyı ifade etmektedir.**
- **Mavi çizgi ile kırmızı çizgi arasındaki alan başarı oranımızdır.**