Meme Kanseri Teşhisi Projesi

GitHub : https://github.com/edanurarslan/Meme-Kanseri-Tespiti

Meme Kanseri Nedir?

Meme kanseri , meme dokusundaki hücrelerin kontrolsüz bir şekilde büyümesi ve çoğalması sonucu oluşan bir kanser türüdür.

Meme kanserinin nedenleri arasında genetik faktörler, hormonal değişimler, yaşam tarzı ve çevresel etmenler bulunur, ayrıca bazı gen mutasyonları meme kanseri riskini artırır.

Tanı ve Erken Teşhis

Dünyada kadınlar arasında en sık görülen kanser türlerinden biridir. Erken teşhis ile tedavi şansı oldukça yüksektir.

- Kendi Kendine Muayene: Düzenli yapılması durumunda erken belirtiler fark edilebilir.
- Mammografi: 40 yaş üstü kadınlarda yıllık tavsiye edilen tarama yöntemidir.
- Ultrason ve MR: Mammografi sonrası ek bilgi gerekirse kullanılır.
- Biyopsi: Şüpheli dokudan örnek alınması.

Korunma ve Önleme

1. Sağlıklı Yaşam Tarzı: Dengeli beslenme, düzenli egzersiz, alkol tüketimini sınırlama.
2. Düzenli Kontroller: Doktor önerisine göre düzenli mammografi ve muayeneler.
3. Genetik Danışmanlık: Aile öyküsü varsa genetik test ve danışmanlık.
4. Risk Azaltıcı Cerrahi: Yüksek riskli kişilerde profilaktik mastektomi.

Makine Öğrenmesi Projesi

Veri Setini Tanıma

```
import pandas as pd

# Veri setini yükleme
file_path = "breast-cancer.csv"
data = pd.read_csv(file_path)

# İlk birkaç satıra göz atma
data.head(10)
```

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean
0	842302	M	17.99	10.38	122.80	1001.0
1	842517	M	20.57	17.77	132.90	1326.0
2	84300903	M	19.69	21.25	130.00	1203.0
3	84348301	M	11.42	20.38	77.58	386.1
4	84358402	M	20.29	14.34	135.10	1297.0
5	843786	M	12.45	15.70	82.57	477.1
6	844359	M	18.25	19.98	119.60	1040.0
7	84458202	M	13.71	20.83	90.20	577.9
8	844981	M	13.00	21.82	87.50	519.8
9	84501001	M	12.46	24.04	83.97	475.9

Özellikler

```python
# Veri setinin genel bilgilerini görüntüleme
data.info()

# Temel istatistiksel özet
statistical_summary = data.describe()
statistical_summary
```

```
---  ------                   --------------  -----
0    id                       569 non-null    int64
1    diagnosis                569 non-null    objec
2    radius_mean              569 non-null    float
3    texture_mean             569 non-null    float
4    perimeter_mean           569 non-null    float
5    area_mean                569 non-null    float
6    smoothness_mean          569 non-null    float
7    compactness_mean         569 non-null    float
8    concavity_mean           569 non-null    float
9    concave points_mean      569 non-null    float
10   symmetry_mean            569 non-null    float
11   fractal_dimension_mean   569 non-null    float
12   radius_se                569 non-null    float
13   texture_se               569 non-null    float
14   perimeter_se             569 non-null    float
15   area_se                  569 non-null    float
16   smoothness_se            569 non-null    float
17   compactness_se           569 non-null    float
18   concavity_se             569 non-null    float
19   concave points_se        569 non-null    float
20   symmetry_se              569 non-null    float
21   fractal_dimension_se     569 non-null    float
22   radius_worst             569 non-null    float
23   texture_worst            569 non-null    float
24   perimeter_worst          569 non-null    float
25   area_worst               569 non-null    float
26   smoothness_worst         569 non-null    float
27   compactness_worst        569 non-null    float
28   concavity_worst          569 non-null    float
29   concave points_worst     569 non-null    float
30   symmetry_worst           569 non-null    float
31   fractal_dimension_worst  569 non-null    float
dtypes: float64(30), int64(1), object(1)
```

Veri Seti İstatistiksel Verileri

```
# Veri setinin genel bilgilerini görüntüleme
data.info()

# Temel istatistiksel özet
statistical_summary = data.describe()
statistical_summary
```

	id	radius_mean	texture_mean	perimeter_mean
count	5.690000e+02	569.000000	569.000000	569.000000
mean	3.037183e+07	14.127292	19.289649	91.969033
std	1.250206e+08	3.524049	4.301036	24.298981
min	8.670000e+03	6.981000	9.710000	43.790000
25%	8.692180e+05	11.700000	16.170000	75.170000
50%	9.060240e+05	13.370000	18.840000	86.240000
75%	8.813129e+06	15.780000	21.800000	104.100000
max	9.113205e+08	28.110000	39.280000	188.500000

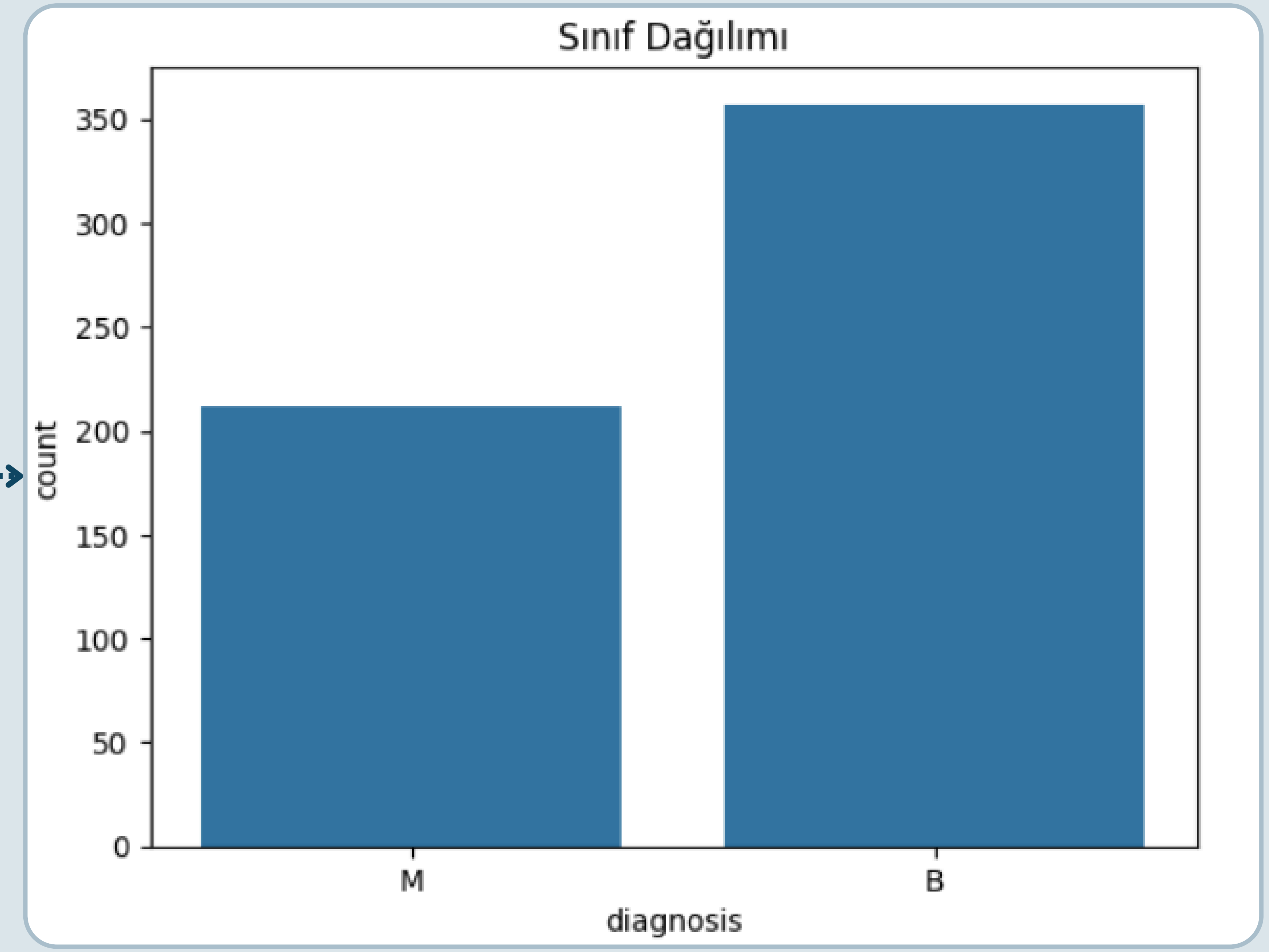
Eksik Veri Kontrolü

```python
# Eksik değer kontrolü
missing_values = data.isnull().sum()

missing_values
```

```
id                         0
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     0
radius_se                  0
texture_se                 0
perimeter_se               0
area_se                    0
smoothness_se              0
compactness_se             0
concavity_se               0
concave points_se          0
symmetry_se                0
fractal_dimension_se       0
radius_worst               0
texture_worst              0
perimeter_worst            0
area_worst                 0
smoothness_worst           0
compactness_worst          0
concavity_worst            0
concave points_worst       0
symmetry_worst             0
fractal_dimension_worst    0
dtype: int64
```

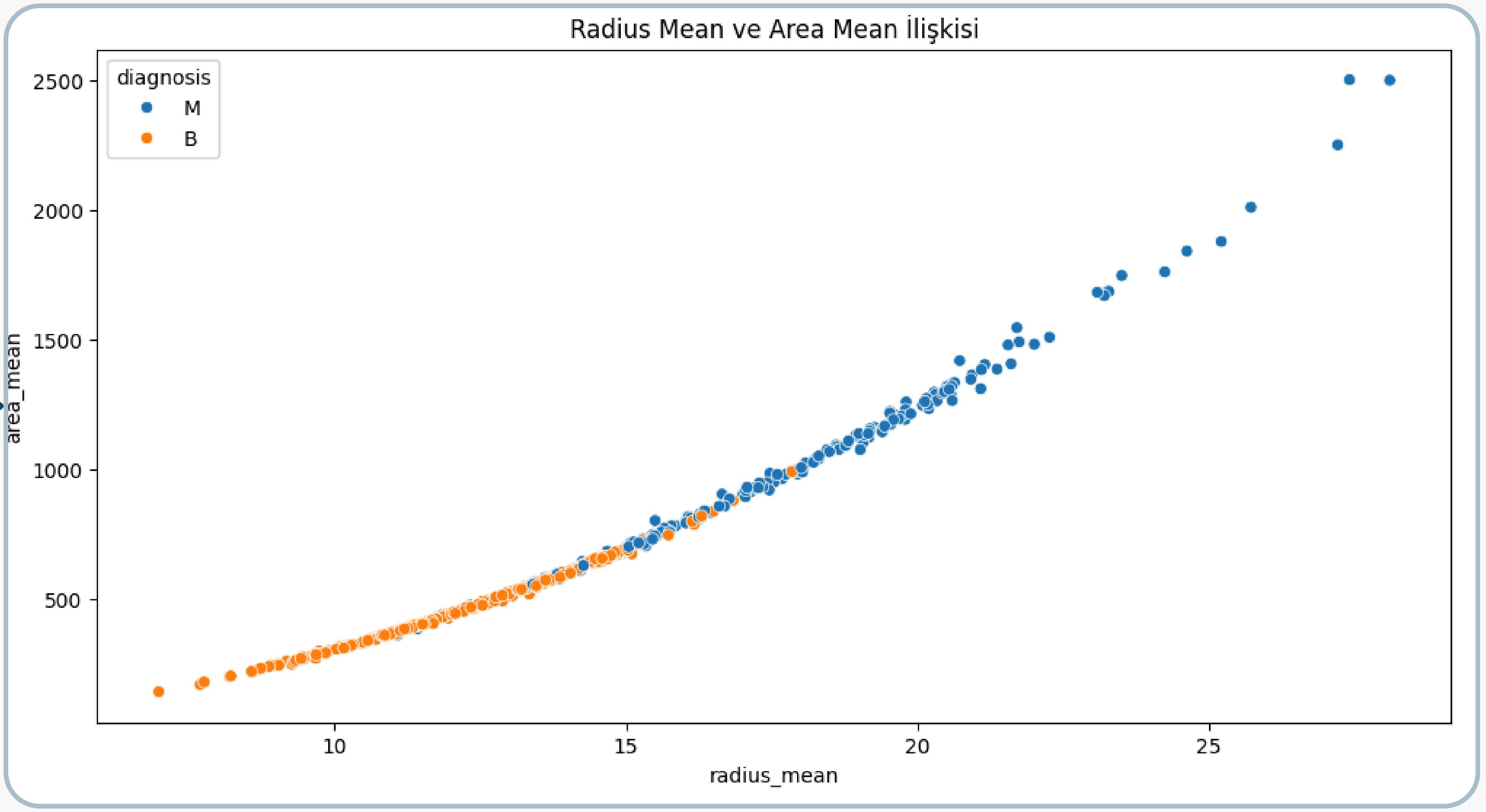
Sınıf Dağılımı İncelemesi

```
#Sınıf dağılımlarının görselleştirilmesi
sns.countplot(data=df, x='diagnosis')
plt.title('Sınıf Dağılımı')
plt.show()

#212 M(Malignant), 357 B(Benign)
#Malignant : Kötü huylu Benign : İyi huylu
```



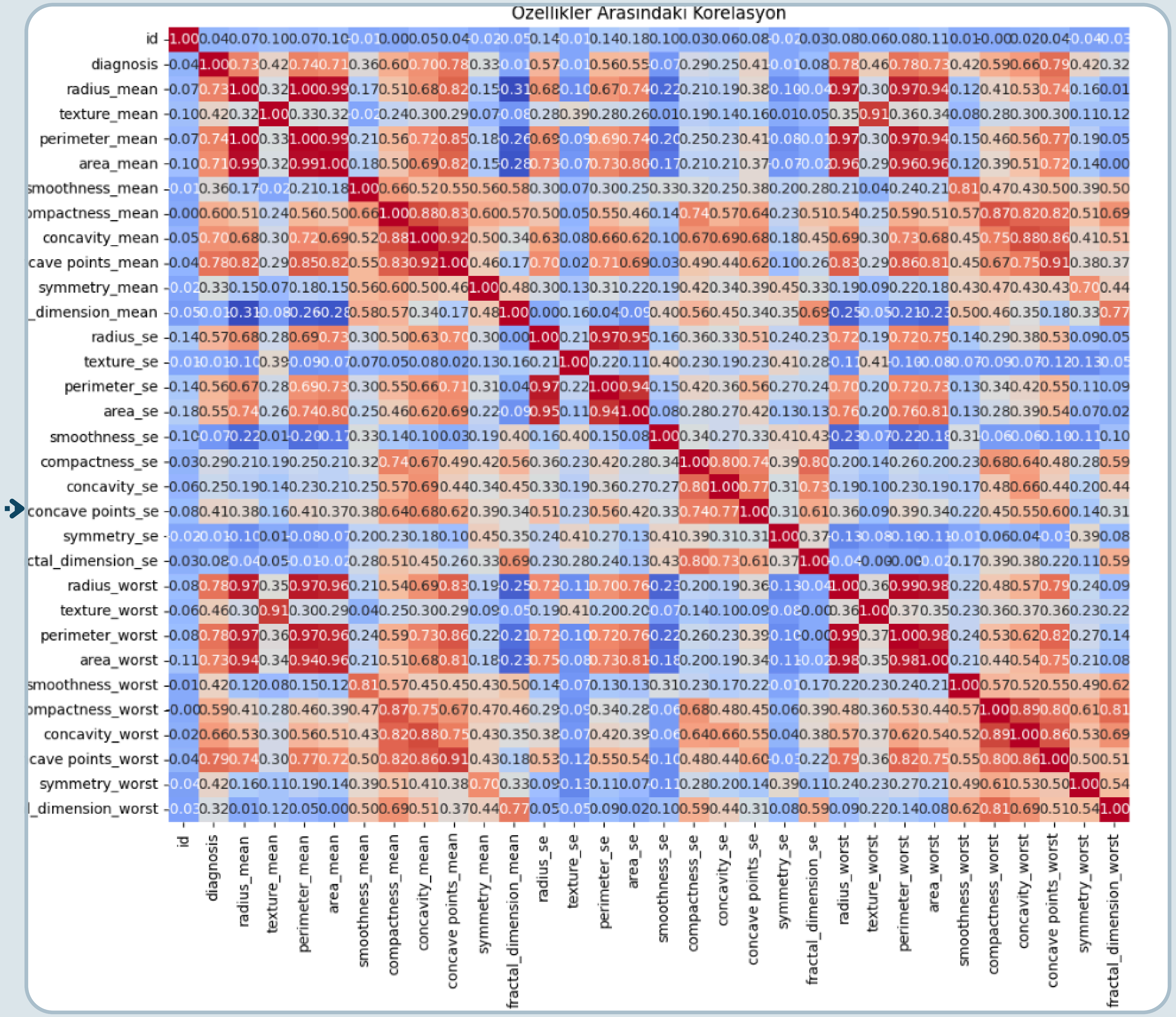
Özellik İlişkisi İncelemesi

```
plt.figure(figsize=(12, 6))
sns.scatterplot(data=df, x='radius_mean', y='area_mean', hue='diagnosis')
plt.title('Radius Mean ve Area Mean İlişkisi')
plt.show()
```



Korelasyon Matrisi

```python
# 'diagnosis' sütununu sayısal verilere dönüştürme
df['diagnosis'] = df['diagnosis'].map({'M': 1, 'B': 0})

# Sayısal sütunları seçme
df_numeric = df.select_dtypes(include=[np.number])

# Korelasyon matrisini hesaplama
correlation_matrix = df_numeric.corr()

# Korelasyon matrisini görselleştirme
plt.figure(figsize=(15, 10))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt='.2f')
plt.title('Özellikler Arasındaki Korelasyon')
plt.show()
```



Model Eğitimi Süreci

1) Veri Setini Hazırlama ve Ölçekleme

```
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.preprocessing import StandardScaler

# Bağımlı ve bağımsız değişkenlerin seçimi
X = df.drop(columns=['id', 'diagnosis'])  # id sütunu çıkarıldı, diagnosis hedef değişken olarak belirlendi
y = df['diagnosis']  # hedef değişken

# Veri setini eğitim ve test olarak ayırma
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Özellikleri ölçeklendirme
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Test seti : %20 Eğitim seti : %80

2) Model Seçimi

a) SVM

```
from sklearn.svm import SVC
#Destek Vektör Makineleri (SVM)

svm_model = SVC()
svm_model.fit(X_train, y_train)

# Tahmin ve değerlendirme
y_pred_svm = svm_model.predict(X_test)
accuracy_svm = accuracy_score(y_test, y_pred_svm)
print("SVM Doğruluk Skoru:", accuracy_svm)
print(classification_report(y_test, y_pred_svm))
```

```
SVM Doğruluk Skoru: 0.9824561403508771
              precision    recall  f1-score   support

           B       0.97      1.00      0.99        71
           M       1.00      0.95      0.98        43

    accuracy                           0.98       114
   macro avg       0.99      0.98      0.98       114
weighted avg       0.98      0.98      0.98       114
```

b) Random Forest

```
from sklearn.ensemble import RandomForestClassifier
#Rastgele Ormanlar (Random Forest)

rf_model = RandomForestClassifier()
rf_model.fit(X_train, y_train)

# Tahmin ve değerlendirme
y_pred_rf = rf_model.predict(X_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print("Random Forest Doğruluk Skoru:", accuracy_rf)
print(classification_report(y_test, y_pred_rf))
```

```
Random Forest Doğruluk Skoru: 0.9649122807017544
              precision    recall  f1-score   support

           B       0.96      0.99      0.97        71
           M       0.98      0.93      0.95        43

    accuracy                           0.96       114
   macro avg       0.97      0.96      0.96       114
weighted avg       0.97      0.96      0.96       114
```

c) XGBoost

```
from xgboost import XGBClassifier
#XGBoost

# 'diagnosis' sütununu sayısal verilere dönüştürme
df['diagnosis'] = df['diagnosis'].map({'M': 1, 'B': 0})

# Hedef değişkenlerin yeniden tanımlanması
y = df['diagnosis']

# Veri setini yeniden ayırma
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Model eğitimi ve tahmin adımlarını tekrarlama
xgb_model = XGBClassifier(random_state=42)
xgb_model.fit(X_train, y_train)

# Tahmin ve değerlendirme
y_pred_xgb = xgb_model.predict(X_test)
accuracy_xgb = accuracy_score(y_test, y_pred_xgb)
print("XGBoost Doğruluk Skoru:", accuracy_xgb)
print(classification_report(y_test, y_pred_xgb))
```

```
XGBoost Doğruluk Skoru: 0.956140350877193
              precision    recall  f1-score   support

           0       0.96      0.97      0.97        71
           1       0.95      0.93      0.94        43

    accuracy                           0.96       114
   macro avg       0.96      0.95      0.95       114
weighted avg       0.96      0.96      0.96       114
```

d) Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
#Karar Ağaçları (Decision Tree)

dt_model = DecisionTreeClassifier(random_state=42)
dt_model.fit(X_train, y_train)

# Tahmin ve değerlendirme
y_pred_dt = dt_model.predict(X_test)
accuracy_dt = accuracy_score(y_test, y_pred_dt)
print("Decision Tree Doğruluk Skoru:", accuracy_dt)
print(classification_report(y_test, y_pred_dt))
```

```
Decision Tree Doğruluk Skoru: 0.9473684210526315
              precision    recall  f1-score   support

           0       0.96      0.96      0.96        71
           1       0.93      0.93      0.93        43

    accuracy                           0.95       114
   macro avg       0.94      0.94      0.94       114
weighted avg       0.95      0.95      0.95       114
```

e) KNN

```
from sklearn.neighbors import KNeighborsClassifier
#K-En Yakın Komşu (KNN)

knn_model = KNeighborsClassifier()
knn_model.fit(X_train, y_train)

# Tahmin ve değerlendirme
y_pred_knn = knn_model.predict(X_test)
accuracy_knn = accuracy_score(y_test, y_pred_knn)
print("KNN Doğruluk Skoru:", accuracy_knn)
print(classification_report(y_test, y_pred_knn))
```

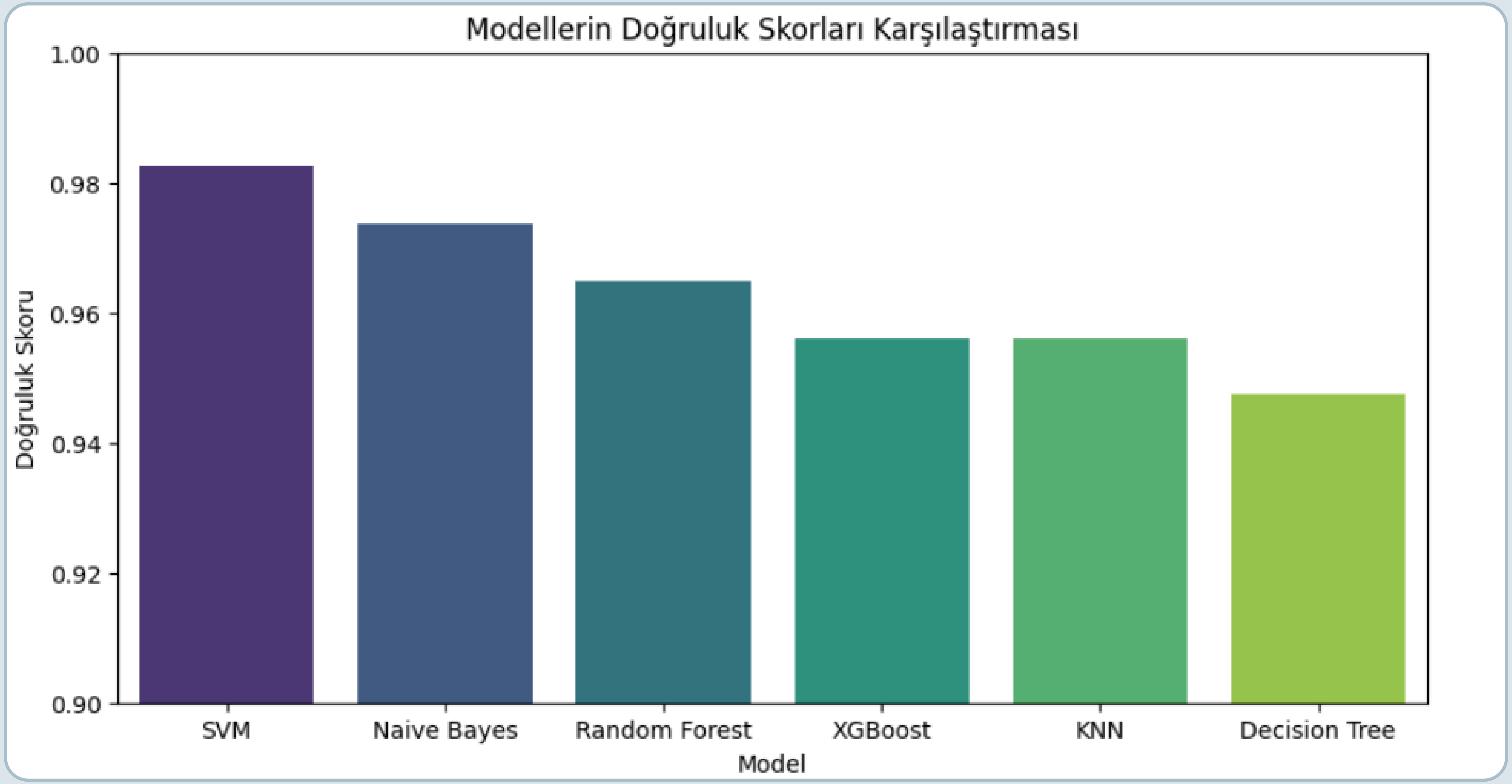
```
KNN Doğruluk Skoru: 0.956140350877193
              precision    recall  f1-score   support

           0       0.93      1.00      0.97        71
           1       1.00      0.88      0.94        43

    accuracy                           0.96       114
   macro avg       0.97      0.94      0.95       114
weighted avg       0.96      0.96      0.96       114
```

f) Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
#Naive Bayes

nb_model = GaussianNB()
nb_model.fit(X_train, y_train)

# Tahmin ve değerlendirme
y_pred_nb = nb_model.predict(X_test)
accuracy_nb = accuracy_score(y_test, y_pred_nb)
print("Naive Bayes Doğruluk Skoru:", accuracy_nb)
print(classification_report(y_test, y_pred_nb))
```

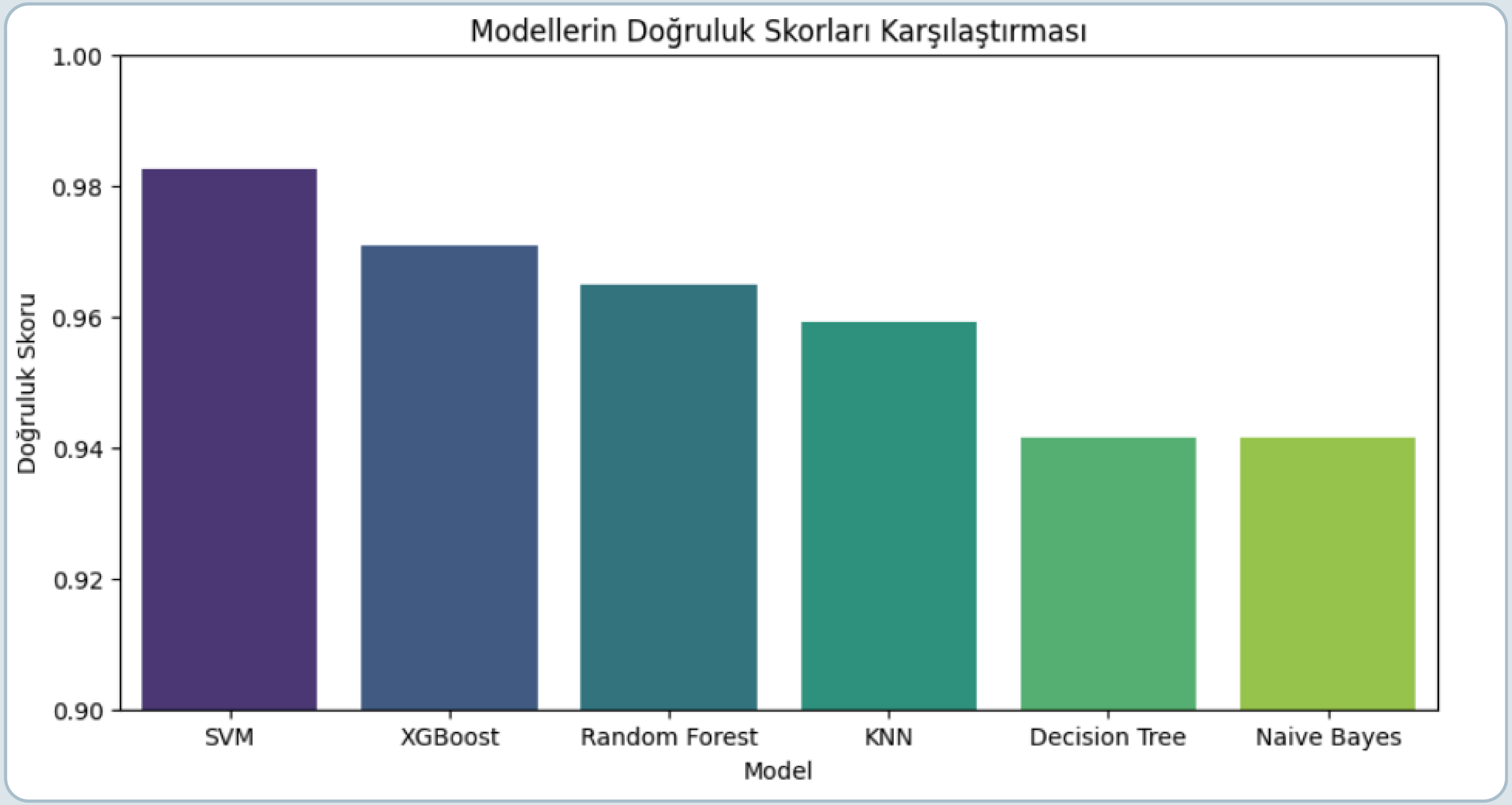
```
Naive Bayes Doğruluk Skoru: 0.9736842105263158
              precision    recall  f1-score   support

           0       0.96      1.00      0.98        71
           1       1.00      0.93      0.96        43

    accuracy                           0.97       114
   macro avg       0.98      0.97      0.97       114
weighted avg       0.97      0.97      0.97       114
```

3) Modellerin Karşılaştırılması



(test_size = 0.2 için)

Seçilen Model : SVM



(test_size = 0.3 için)

Seçilen Model : SVM

Hiperparametre Optimizasyonu Örneği

Optimize edilmiş model :

```
from sklearn.model_selection import GridSearchCV, RandomizedSearchCV
# Örnek: XGBoost Hiperparametre Optimizasyonu

# Hiperparametreler için bir grid oluşturma
xgb_param_grid = {
    'n_estimators': [100, 200, 300],
    'learning_rate': [0.01, 0.1, 0.2],
    'max_depth': [3, 5, 7],
    'subsample': [0.7, 0.8, 0.9],
    'colsample_bytree': [0.7, 0.8, 0.9]
}

# RandomizedSearchCV ile optimizasyon
xgb_random_search = RandomizedSearchCV(estimator=xgb_model, param_distributions=xgb_param_grid, n_iter=50, cv=5, n_jobs=-1, verbose=2, random_state=42)
xgb_random_search.fit(X_train, y_train)

# En iyi hiperparametreler
print("En iyi XGBoost hiperparametreleri: ", xgb_random_search.best_params_)
print("En iyi XGBoost doğruluk skoru: ", xgb_random_search.best_score_)
```

```
Fitting 5 folds for each of 50 candidates, totalling 250 fits
En iyi XGBoost hiperparametreleri:  {'subsample': 0.8, 'n_estimators': 200, 'max_depth': 5, 'learning_rate': 0.2, 'colsample_bytree': 0.7}
En iyi XGBoost doğruluk skoru:  0.9802197802197803
```

Model Değerlendirme 1

```
best_model = svm_model
```

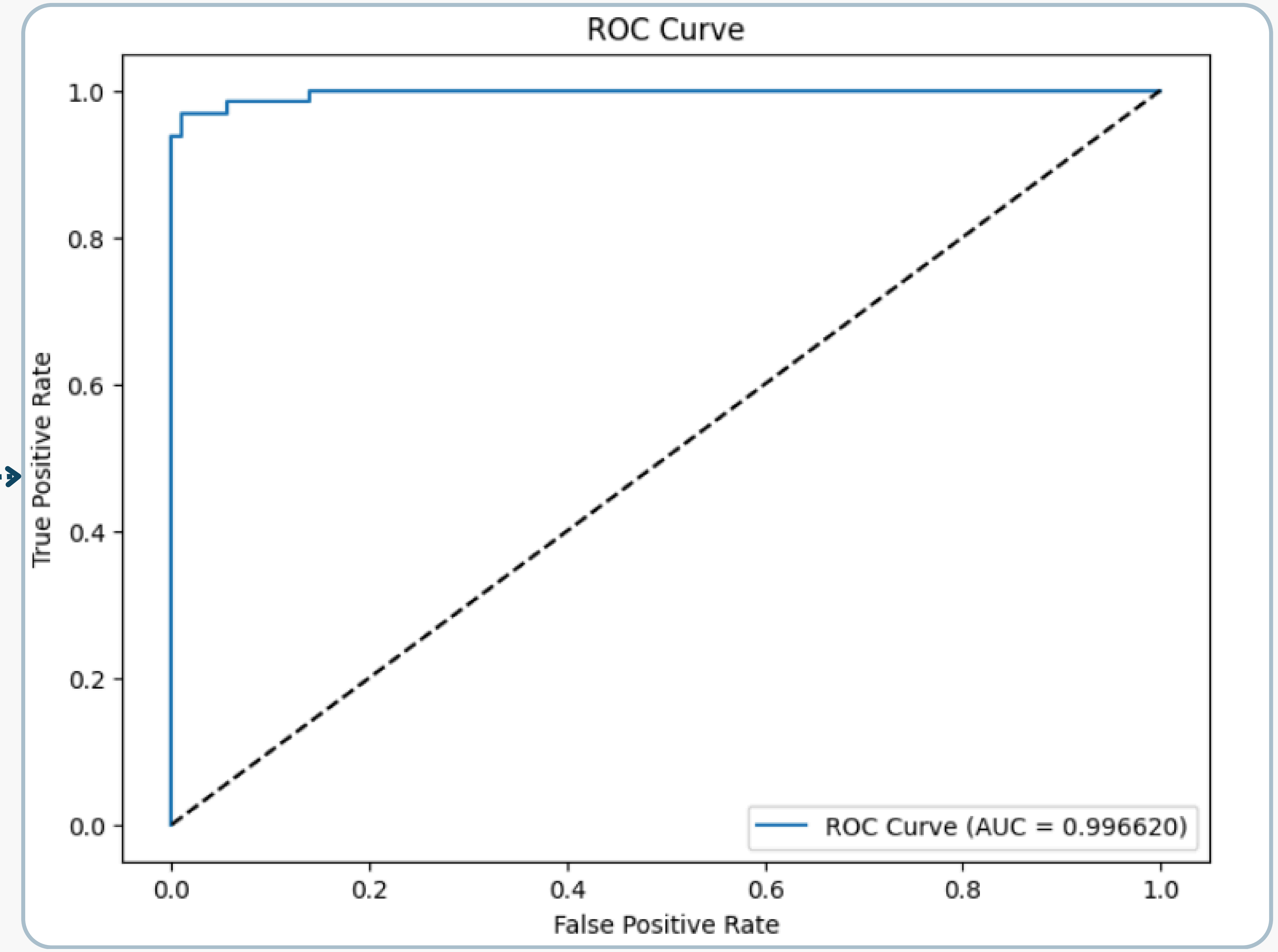
```
# K-Fold Cross Validation
k = 10
cv_scores = cross_val_score(best_model, X, y, cv=k)

# K-Fold Cross Validation sonuçlarının ortalamasını ve standart sapmasını hesaplama
print(f"{k}-Fold Cross Validation Ortalama Doğruluk Skoru: {cv_scores.mean():.6f}")
print(f"{k}-Fold Cross Validation Standart Sapma: {cv_scores.std():.6f}")
```

```
10-Fold Cross Validation Ortalama Doğruluk Skoru: 0.913878
10-Fold Cross Validation Standart Sapma: 0.028787
```

Model Değerlendirme 2

```python
# Precision, Recall, F1 Score hesaplama
precision = precision_score(y_test, y_pred, pos_label='M')
recall = recall_score(y_test, y_pred, pos_label='M')
f1 = f1_score(y_test, y_pred, pos_label='M')

print(f"Precision: {precision:.6f}")
print(f"Recall: {recall:.6f}")
print(f"F1 Score: {f1:.6f}")

# ROC-AUC Score ve ROC Curve
if hasattr(best_model, "predict_proba"):
    y_pred_prob = best_model.predict_proba(X_test)[:, 1]
elif hasattr(best_model, "decision_function"):
    y_pred_prob = best_model.decision_function(X_test)
else:
    raise AttributeError("Model predict_proba veya decision_function yöntemlerine sahip değil.")

roc_auc = roc_auc_score(y_test.map({'B': 0, 'M': 1}), y_pred_prob)
fpr, tpr, thresholds = roc_curve(y_test.map({'B': 0, 'M': 1}), y_pred_prob)

plt.figure(figsize=(8, 6))
plt.plot(fpr, tpr, label=f'ROC Curve (AUC = {roc_auc:.6f})')
plt.plot([0, 1], [0, 1], 'k--')  # Random classifier line
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC Curve')
plt.legend(loc='best')
plt.show()
```

```
Precision: 0.968254
Recall: 0.968254
F1 Score: 0.968254
```



“ERKEN TEŞHİS HAYAT KURTARIR.”

1 - 31 Ekim Farkındalık Ayı

Teşekkürler

GitHub : https://github.com/edanurarslan/Meme-Kanseri-Tespiti