Yapay Zeka Akademisi Bitirme Projesi

1. Problem Tanımı

1. Hangi veri seti seçildi?

Bu çalışmada, Kaggle platformundan alınan “Heart Disease Data” veri seti kullanılmıştır. Veri seti, bireylerin demografik ve tıbbi bilgilerini içererek kalp hastalığına yakalanma durumlarını yansıtır. 14’ten fazla özelliği içeren bu veri seti, sınıflandırma problemleri için sıklıkla tercih edilmektedir.

Veri seti linki: https://www.kaggle.com/datasets/redwankarimsony/heart-disease-data/data

2. Problemin Amacı ve Hedef Değişken

Bu projenin amacı, bireylere ait sağlık ve yaşam tarzı verilerini kullanarak kişinin kalp hastalığına sahip olup olmadığını tahmin etmektir.

Hedef değişken: num,

- num = 0 → Kişide kalp hastalığı yok, num = 1, 2, 3, 4 → Kişide kalp hastalığı var

- Hedef değişken : target olarak güncellenmiştir. target = 0 (Kalp hastalığı yok), target = 1 (Kalp hastalığı var)

3. Tahminin Pratik Önemi

Kalp hastalıkları, dünya çapında önde gelen ölüm nedenlerinden biridir. Erken teşhis, tedavi sürecini hızlandırır ve ölüm riskini azaltır. Bu nedenle makine öğrenimi algoritmalarıyla geliştirilecek modeller, bireylerin sağlık verilerinden erken teşhis yapılmasına katkı sağlayabilir.

2. Veri Ön İşleme

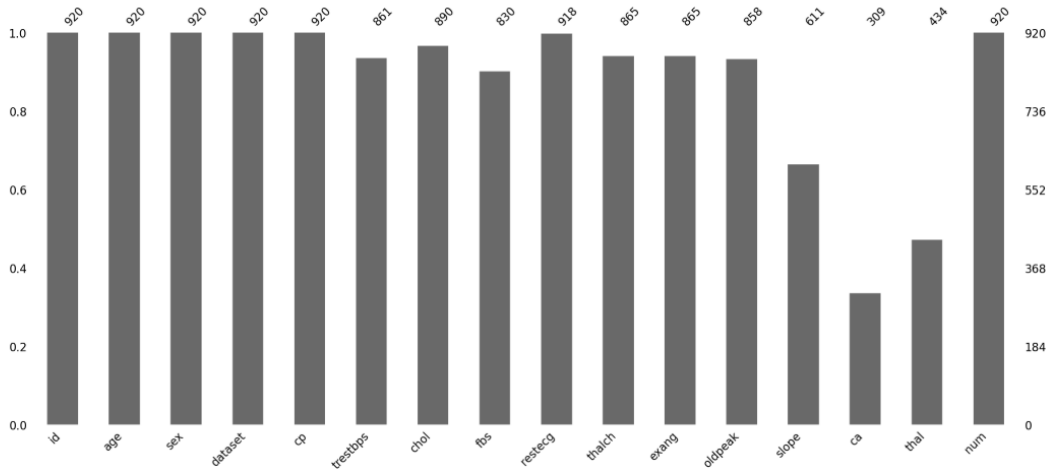
Veri setinde aşağıdaki ön işleme adımları uygulanmıştır:

- Eksik veriler kontrol edilmiştir.

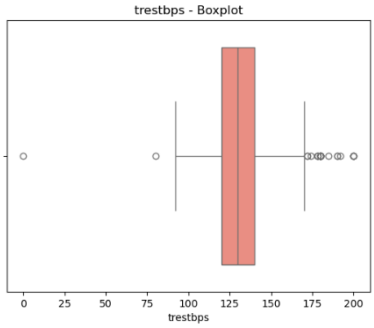
Eksik Veri Görselleştirme

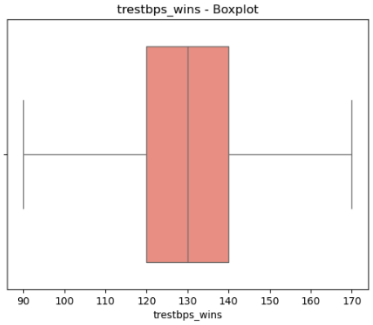
Eksik verilerin değerlerini daha net görmek adına missingno grafiğini kullanarak görselleştirme işlemi gerçekleştirelim.

```
[10]: # missingno ile eksik veri bar grafiği
      msno.bar(df)          # missingno kütüphanesine yer alan bar grafik ile verinin dağılımını görselleştir
      plt.show()            # ekranda gösterme işlemi
```

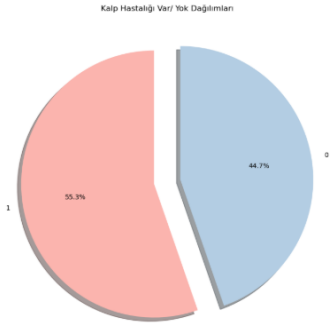


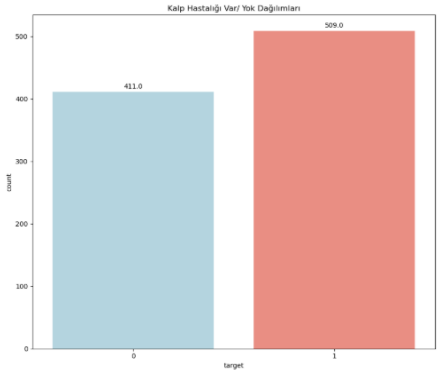
- Outlier (aykırı değer) kontrolü yapılmış, Winsorizer ile uç değer baskılanmıştır.



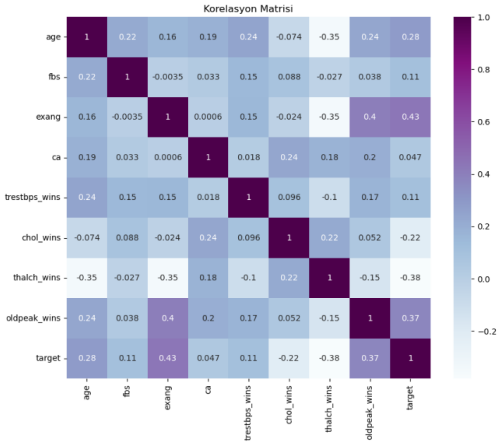


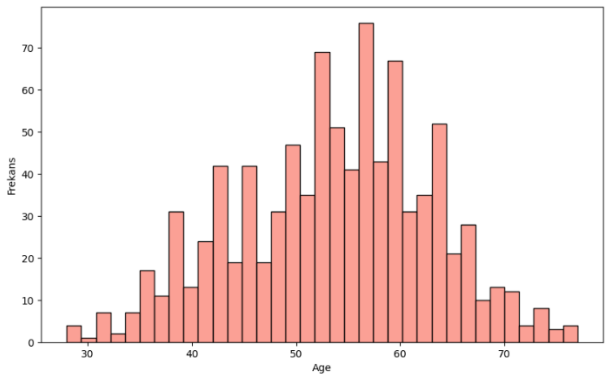
- Hedef değişkenin dağılımı incelenmiştir. Veri dengesizliği problemi gözlenmemiştir, dolayısıyla SMOTE gibi tekniklere ihtiyaç duyulmamıştır.





- Korelasyonlar, dağılımlar veya kategorik etkileşimlere yer verilmiştir.





- Kategorik değişkenler label encoding ve one-hot encoding ile dönüştürülmüştür.

Label Encoder

[64]:
```
# Gerekli kütüphanelerin yüklenmesi

from sklearn.preprocessing import LabelEncoder
```

[65]:
```
le = LabelEncoder()                 # LabelEncoder nesnesi oluştur

df['sex_label'] = le.fit_transform(df['sex'])    # 'sex' sütununu sayısal değerlere dönüştür ve yeni sütun olarak ekle
df['fbs_label'] = le.fit_transform(df['fbs'])    # 'fbs' sütununu sayısal değerlere dönüştür ve yeni sütun olarak ekle
df['exang_label'] = le.fit_transform(df['exang'])# 'exang' sütununu sayısal değerlere dönüştür ve yeni sütun olarak ekle
```

Yorum:

- sex_label değişkeni için >> Male = 1, Female = 0
- fbs_label değişkeni için >> True = 1, False = 0
- exang_label değişkeni için >> True = 1, False = 0

Görüldüğü üzere, Label Encoder kullanarak gender sütunundaki kategorik değerleri sayısal değerlere dönüştürdük.

One-Hot Encoder

[66]:
```
# one-hot-encoding yöntemi ile kategorik veriyi sayısala çeviriyoruz. drop_first ile ilk sütunu düşürüyoruz.

df = pd.get_dummies(df, columns=['cp', 'restecg', 'slope', 'thal'], drop_first=True, dtype='int64')
```

- Eğitim/test ayrımı (train_test_split, 70-30 ya da 80-20) gerçekleştirilmiştir. Sayısal veriler StandardScaler ile normalize edilmiştir. Modelleme için gerekli hazırlıklar tamamlanmıştır.

```
[76]: # Eğitim/test verilerini ayırma

X_train, X_test, y_train, y_test = train_test_split(
    X, y,                   # Özellikler ve hedef değişken
    train_size=0.70,        # %70 eğitim, %30 test verisi olacak şekilde böl
    random_state=42         # Sonuçların tekrarlanabilir olması için sabit rastgelelik
)

# StandardScaler ile veriyi ölçeklendirme (standardizasyon)
scaler = StandardScaler()          # StandardScaler nesnesi oluştur

X_train_std = scaler.fit_transform(X_train)  # Eğitim verisini öğren ve dönüştür
X_test_std = scaler.transform(X_test)        # Test verisini aynı dönüşümle dönüştür (fit etmiyoruz!)
```

3. Modelleme ve Sonuçların Değerlendirilmesi

Projede aşağıdaki makine öğrenmesi modelleri kullanılmıştır:

- Logistic Regression

```
[78]: # Modeli eğitiyoruz (öğrenme tamamlanacak)

log_model = LogisticRegression(
    max_iter=1000,           # Maksimum iterasyon sayısı (modelin öğrenme için izin verilen adım sayısı)
    class_weight='balanced'  # Dengesiz sınıflar için ağırlıklandırma yapar
)

log_model.fit(X_train_std, y_train)  # Modeli eğitim verisi ile eğit
```

```
[78]: LogisticRegression
LogisticRegression(class_weight='balanced', max_iter=1000)
```

- Random Forest

```
[212]: # Random Forest modelini eğitim verisi ile eğitiyoruz

rf_model.fit(X_train_rand_scaled, y_train_rand)  # Modeli özellikler ve hedef değişken ile öğren
```

```
[212]: RandomForestClassifier
RandomForestClassifier(max_depth=5, min_samples_leaf=4, min_samples_split=10,
                       n_estimators=200, random_state=42)
```

- KNN (K-Nearst Neighb ors)

```
[226]: # En iyi k değerini bul ve final modeli oluştur

best_k = k_values[np.argmax(mean_test_score)]   # En yüksek ortalama doğruluk değerine sahip k'yı seç
print("En iyi k değeri:", best_k)               # En iyi k değerini yazdır

knn_final = KNeighborsClassifier(
    n_neighbors=best_k,
    weights='uniform'                           # Komşulara eşit ağırlık ver
)

knn_final.fit(X_train_scaled_knn, y_train_knn)  # Final modeli eğitim verisi ile eğit
```

```
En iyi k değeri: 10
```

```
[226]: KNeighborsClassifier
KNeighborsClassifier(n_neighbors=10)
```

5.1. Hangi model daha iyi performans verdi?

Genel değerlendirme: Random Forest modeli en iyi genel performansı göstermiştir.

Model	Accuracy	Precision	Recall	F1 Score	AUC	TN	FP	FN	TP
Logistic Regression	0.80	0.84	0.80	0.82	0.87	96	24	31	125
Random Forest	**0.83**	0.83	**0.86**	**0.85**	**0.90**	97	26	**22**	**131**
KNN	0.80	**0.87**	0.77	0.82	0.89	**102**	**18**	36	120

- Accuracy: Random Forest en yüksek doğruluk oranına sahip (0.83).
- F1 Score: Random Forest precision ve recall dengesinde en iyi sonucu verdi (0.85).
- AUC (ROC): Random Forest 0.90 ile en yüksek ayırma gücüne sahip.
- TP ve FN: Random Forest en fazla doğru pozitif (131) ve en az yanlış negatif (22) sayısına sahip, yani pozitif sınıfı en iyi yakalayan model.
- KNN yüksek precision değerine sahip ancak recall düşük; yani pozitifleri biraz daha kaçırıyor.

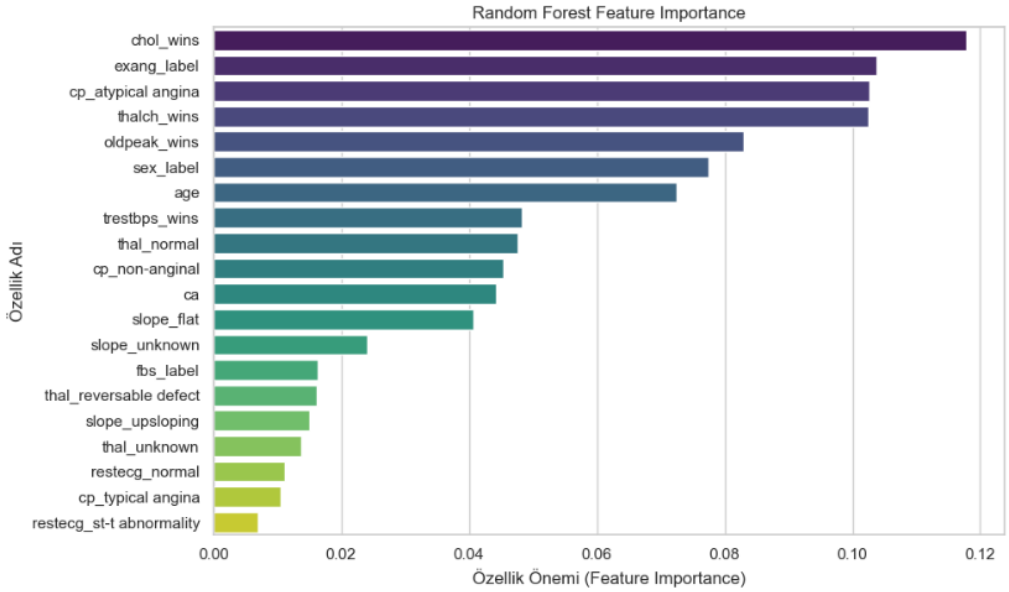
Sonuç: Random Forest modeli genel olarak diğer modellere göre daha iyi performans göstermiştir.

4. Özellik Önem Değerlendirmesi

GridSearchCV ile hiperparametre optimizasyonu gerçekleştirilmiştir. En iyi parametre değerleri ile en iyi sonuca ulaşılmak amaçlanmıştır.

```
[245]: # Modeli eğitelim (fit edelim)

grid_search.fit(X_train, y_train)  # Eğitim verisi üzerinde GridSearchCV ile hiperparametre araması yap ve modeli eğit

Fitting 5 folds for each of 216 candidates, totalling 1080 fits

[245]: GridSearchCV
       best_estimator_: RandomForestClassifier
           RandomForestClassifier

[246]: # En İyi Parametreleri ve Sonuçları Görüntüle

print("En iyi parametreler:")
print(grid_search.best_params_)  # GridSearch sonucu bulunan en iyi hiperparametreler

print("\nEn iyi F1 skoru:")
print(grid_search.best_score_)   # En iyi modelin çapraz doğrulama sonucu F1 skoru

En iyi parametreler:
{'bootstrap': False, 'max_depth': None, 'min_samples_leaf': 2, 'min_samples_split': 10, 'n_estimators': 100}

En iyi F1 skoru:
0.8406192668914908
```

Random Forest modeli ile yapılan özellik önem analizi sonucunda en etkili değişkenler şunlardır:



5. Sonuç ve Çıkarımlar

- Kalp hastalığı tahmini için geliştirilen tüm modeller yüksek doğruluk oranına sahiptir.
- Tüm modeller (%80 ve üzeri doğrulukla) test verisinde başarılı sonuçlar vermiştir.
- Overfitting veya underfitting gözlenmemiştir, yani modeller genelleme yeteneğine sahiptir.
- Random Forest, hem test hem de eğitim skorları açısından en istikrarlı ve dengeli sonuçları verdi.