FET445 Veri Madenciliği

Araba Fiyat Tahmini

Grup : Motor Çetesi

Tarih:25.12.2025

Problem Tanımı

Bu projede, ikinci el araç piyasasında yer alan fiyatların gerçekçi olup olmadığını analiz etmek ve kullanıcılara veriye dayalı bir fiyat tahmini sunmak amaçlanmıştır.

Craigslist.org üzerinden elde edilen ikinci el araç verileri kullanılarak, araçların sayısal ve kategorik özelliklerine dayalı bir makine öğrenmesi modeli geliştirilmiştir.

Problem, araç özelliklerinden yola çıkarak satış fiyatını tahmin etmeyi hedefleyen bir regresyon problemi olarak ele alınmıştır.

Modelin başarımı MAE, RMSE ve R^2 metrikleri ile değerlendirilmiş; düşük tahmin hatası ve yüksek açıklayıcılık elde edilmesi hedeflenmiştir.

Veri Seti

Bu projede, ABD genelinde yayınlanan ikinci el araç ilanlarını içeren Craigslist Cars & Trucks Dataset kullanılmıştır. Veri seti, Craigslist.org üzerinden web scraping yöntemiyle toplanmış gerçek araç ilanlarından oluşmaktadır.

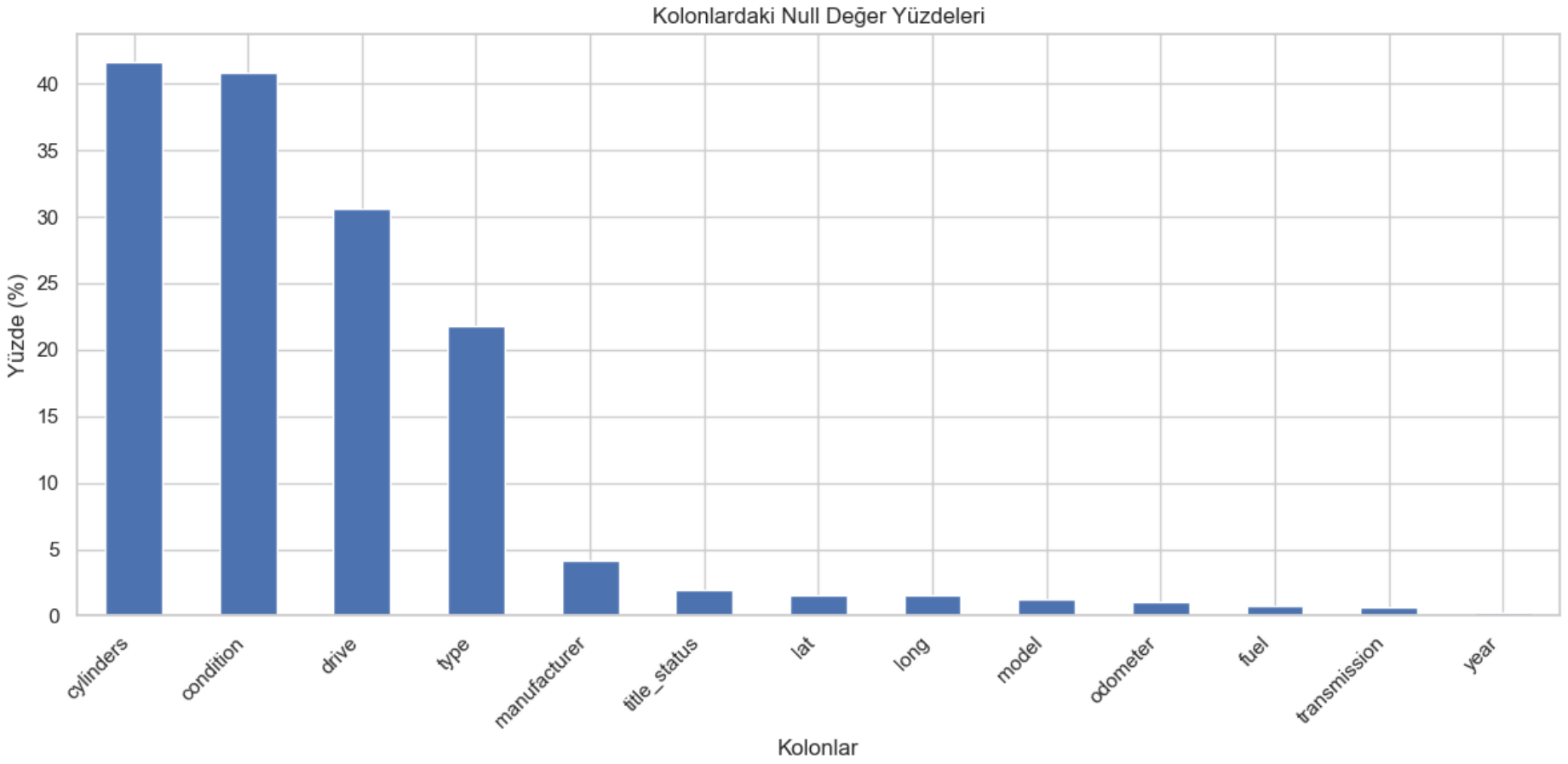
Amaç, araçlara ait sayısal, kategorik ve ordinal özellikleri kullanarak satış fiyatını tahmin etmektir. Farklı araç tipleri, üreticiler ve kullanım durumlarını kapsaması sayesinde veri seti, gerçekçi ve geniş bir piyasa temsili sunmaktadır.

Veri Seti Boyutu

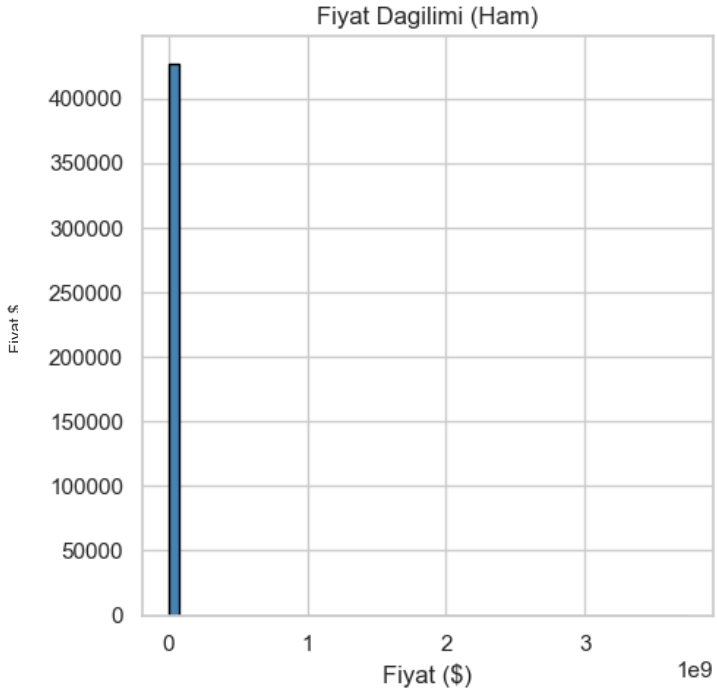
- Toplam satır sayısı: 426.880
- Toplam sütun sayısı: 26
- Dosya boyutu: ~1.45 GB (CSV)

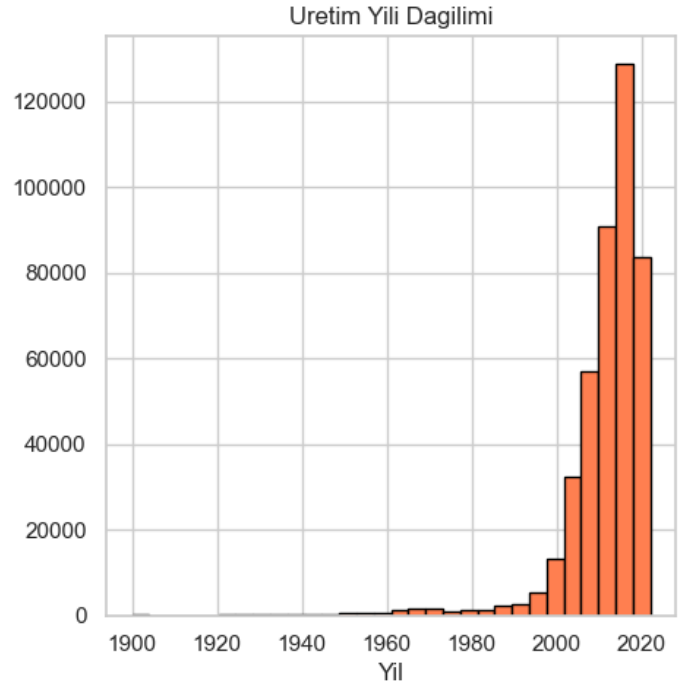
Veri Seti Özellik Tipleri

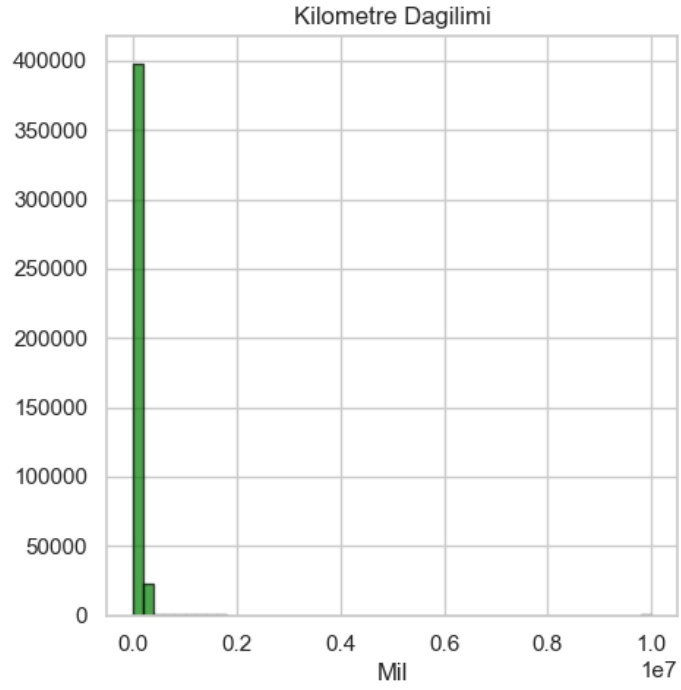
- Sayısal özellikler:
  price (hedef değişken), year, odometer, cylinders, latitude, longitude
- Kategorik özellikler:
  manufacturer, fuel, transmission, drive, type, paint_color, title_status, state, region
- Ordinal özellikler:
  condition

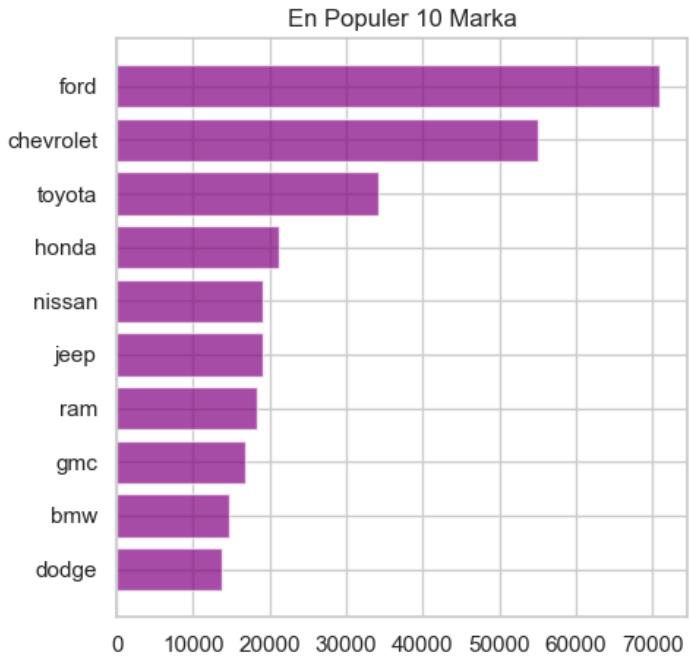


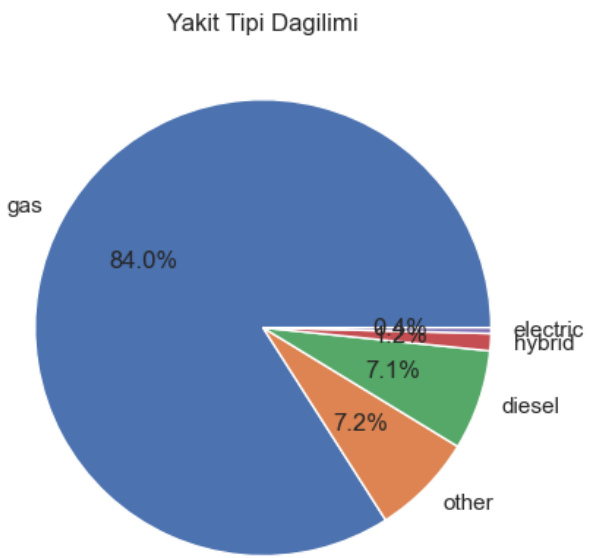
EDA öncesi kolonlardaki null değer yüzde grafiği

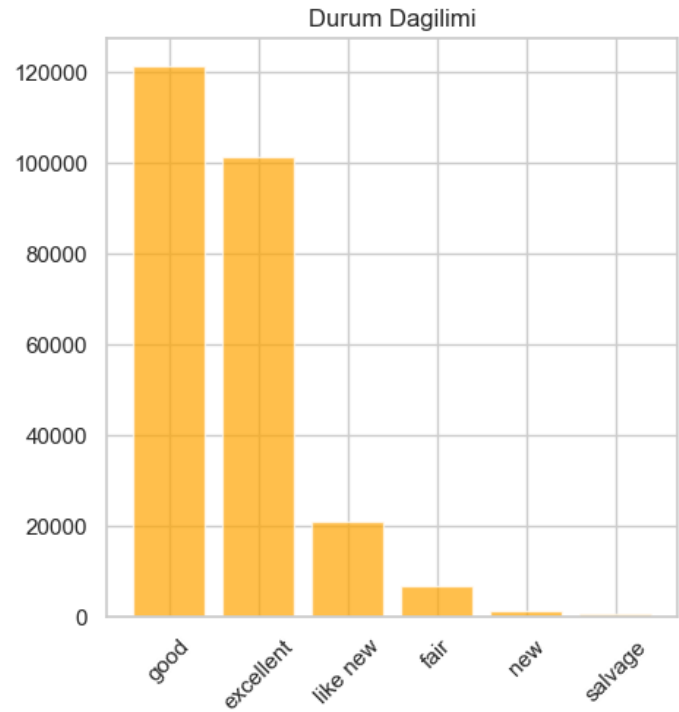

Fiyat Dagilimi (Ham)
Fiyat ($)
1e9

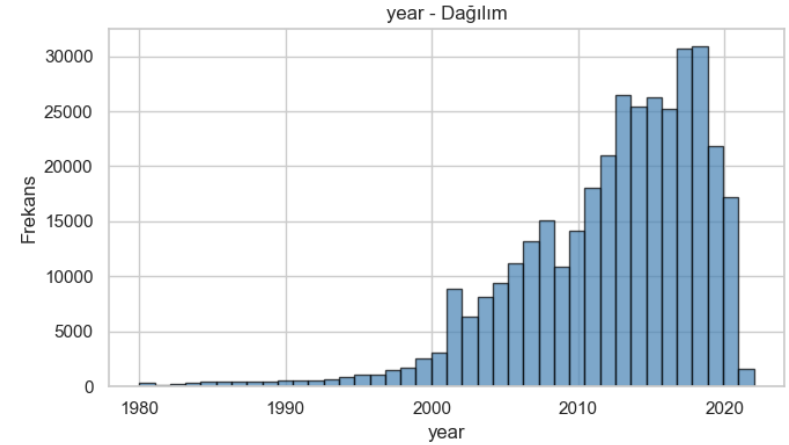

Uretim Yili Dagilimi
Yil

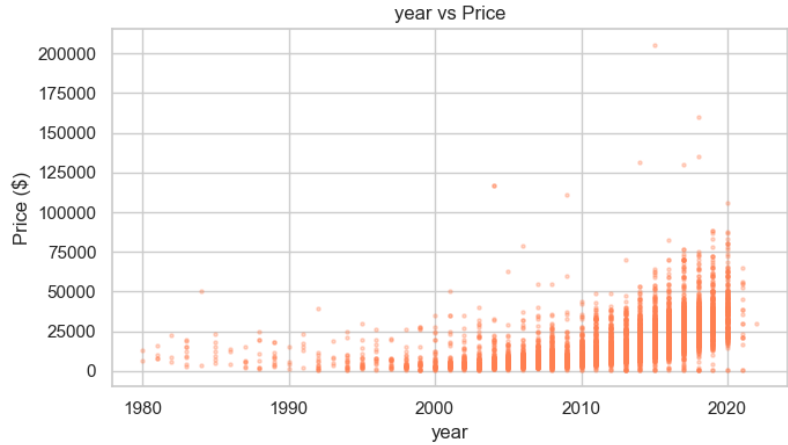

Kilometre Dagilimi
Mil
1e7

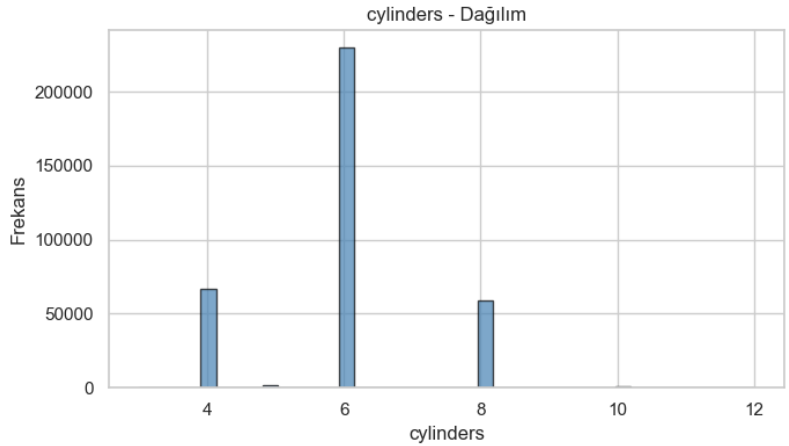

En Populer 10 Marka
ford
chevrolet
toyota
honda
nissan
jeep
ram
gmc
bmw
dodge

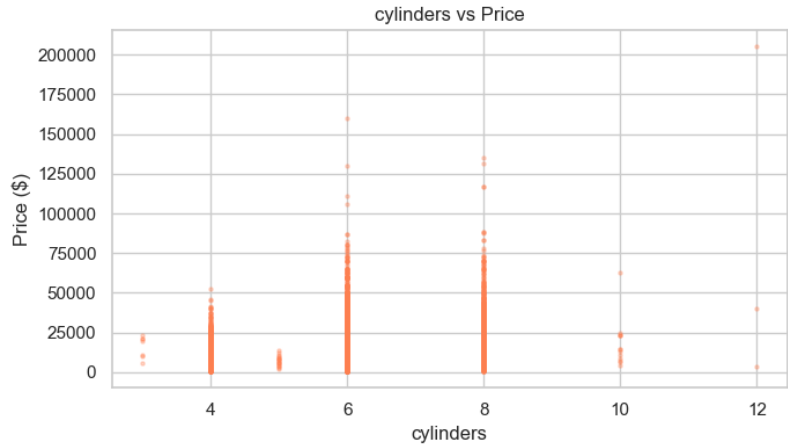

Yakit Tipi Dagilimi
gas
84.0%
electric
hybrid
diesel
7.1%
other
7.2%

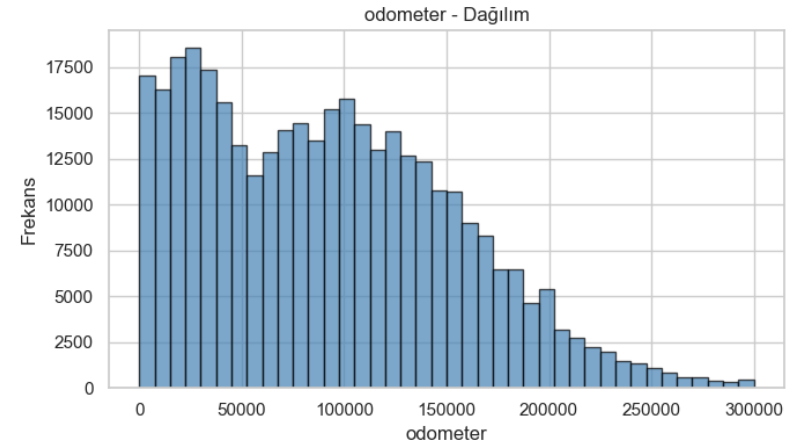

Durum Dagilimi
good
excellent
like new
fair
new
salvage

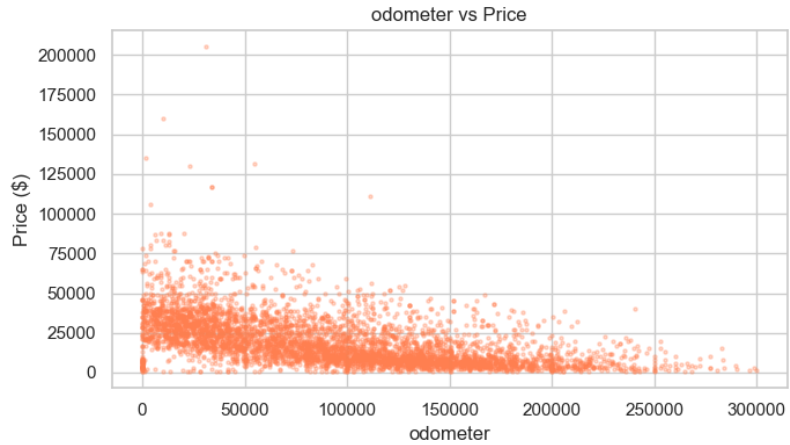

year - Dağılım
Frekans
year
0
5000
10000
15000
20000
25000
30000
1980
1990
2000
2010
2020

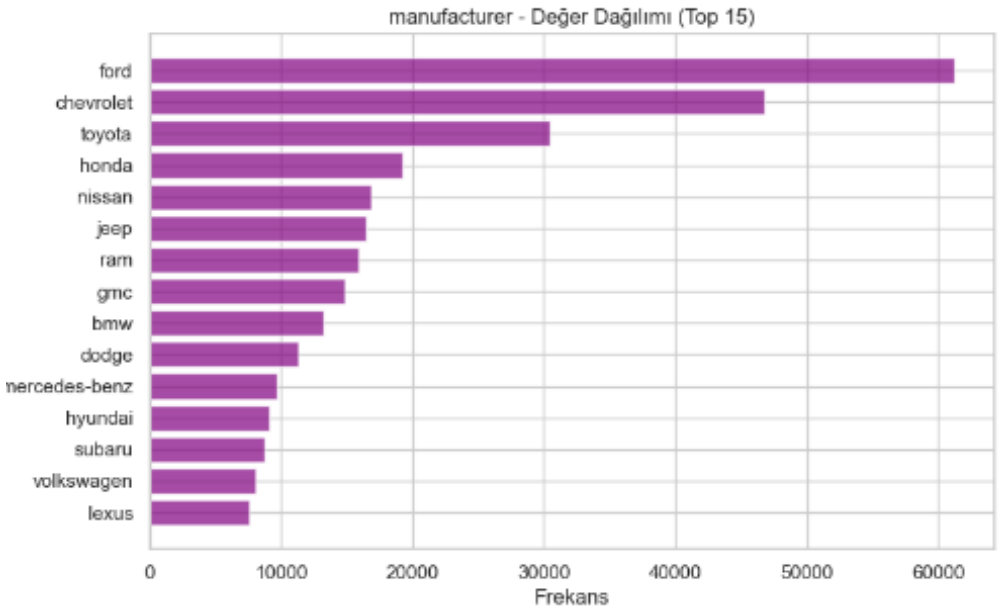

year vs Price
Price ($)
year
0
25000
50000
75000
100000
125000
150000
175000
200000
1980
1990
2000
2010
2020

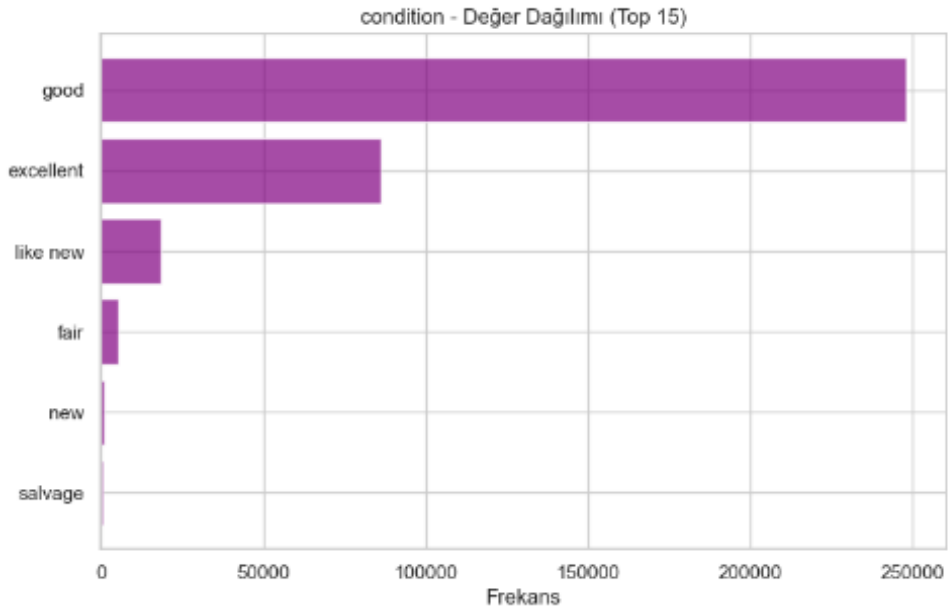

cylinders - Dağılım
Frekans
cylinders
0
50000
100000
150000
200000
4
6
8
10
12

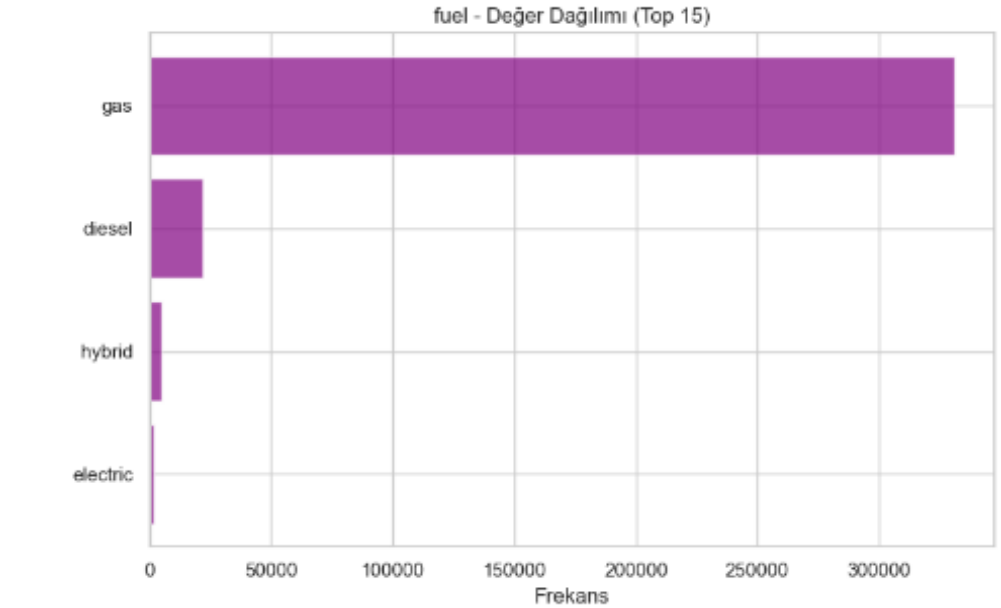

cylinders vs Price
Price ($)
cylinders
0
25000
50000
75000
100000
125000
150000
175000
200000
4
6
8
10
12

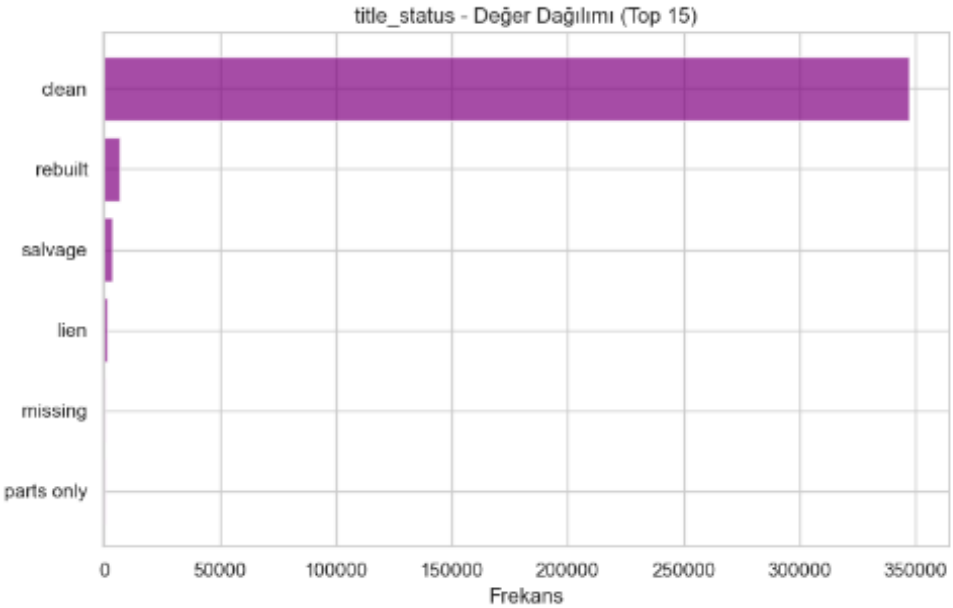

odometer - Dağılım
Frekans
odometer
0
2500
5000
7500
10000
12500
15000
17500
0
50000
100000
150000
200000
250000
300000


odometer vs Price
Price ($)
odometer
0
25000
50000
75000
100000
125000
150000
175000
200000
0
50000
100000
150000
200000
250000
300000


manufacturer - Değer Dağılımı (Top 15)
ford
chevrolet
toyota
honda
nissan
jeep
ram
gmc
bmw
dodge
mercedes-benz
hyundai
subaru
volkswagen
lexus
0
10000
20000
30000
40000
50000
60000
Frekans


condition - Değer Dağılımı (Top 15)
good
excellent
like new
fair
new
salvage
0
50000
100000
150000
200000
250000
Frekans


fuel - Değer Dağılımı (Top 15)
gas
diesel
hybrid
electric
0
50000
100000
150000
200000
250000
300000
Frekans


title_status - Değer Dağılımı (Top 15)
clean
rebuilt
salvage
lien
missing
parts only
0
50000
100000
150000
200000
250000
300000
350000
Frekans

Her bir ekip üyesi farklı feature engineering modelleri + sabit bir baseline model kullanarak farklı feature engineering tekniklerinin direkt olarak R2 , RMSE , MAE metriklerine etkisini test etmiştir. Kullanılan bazı feature engineering teknikleri:

```python
def categorize_engine(cylinders):
    if cylinders <= 4:
        return 'small_engine'
    elif cylinders <= 6:
        return 'medium_engine'
    else:
        return 'large_engine'

df['engine_category'] = df['cylinders'].apply(categorize_engine)
```

```python
def categorize_age(age):
    if age <= 2:
        return 'brand_new'
    elif age <= 5:
        return 'new'
    elif age <= 10:
        return 'mid_age'
    elif age <= 15:
        return 'old'
    else:
        return 'very_old'

df['age_category'] = df['car_age'].apply(categorize_age)
```

```python
# 1) Arac yasi
df['car_age'] = 2025 - df['year']
print('car_age eklendi')

# 2) Yillik km
df['mil_per_year'] = df['odometer'] / df['car_age'].replace(0, 1)
print('mil_per_year eklendi')

# 3) Condition duzeltme (km bazli)
def correct_condition(row):
    odometer = row['odometer']
    if odometer <= 10000: return 'new'
    elif odometer <= 50000: return 'like new'
    elif odometer <= 100000: return 'excellent'
    elif odometer <= 150000: return 'good'
    else: return 'fair'

df['condition'] = df.apply(correct_condition, axis=1)
print('condition km bazli duzeltildi')
```

```
Toplam canonical model sayısı: 665
Toplam model varyant sayısı: 1050

================================================================================
MODEL NORMALİZASYON İŞLEMİ
================================================================================

Ham model NaN sayısı: 4,452
Normalize edilmiş model sayısı: 364,044

Sözlükte olmayan model sayısı: 202,516
Bu modeller 'unknown' olarak işaretlenecek (silinmeyecek)

Final dataframe boyutu: 368,498 satır (veri kaybı yok!)
Model kolonu hazır: 364,044 normalize edilmiş model
  - Bilinen modeller: 165,982
  - Unknown modeller: 202,516

================================================================================
MODEL NORMALİZASYON TAMAMLANDI
================================================================================

================================================================================
UNKNOWN MODEL DEĞERLERİNİ DOLDURMA (Manufacturer'a göre)
================================================================================

Doldurulacak unknown model sayısı: 202,516

Manufacturer bazında en popüler modeller bulundu: 40 manufacturer

Doldurma sonuçları:
  - Doldurulan model sayısı: 202,100
  - Kalan unknown model sayısı: 416
  - Doldurma oranı: 99.8%
  - Kalan 416 unknown değer 'other' olarak işaretlendi

================================================================================
UNKNOWN MODEL DOLDURMA TAMAMLANDI
================================================================================
```

```
=== TYPE DOLDURMA (MODEL BİLGİSİ İLE) ===

Başlangıç type null: 86295
1) Model isminden type çıkarılıyor...
   Sonrası null: 49988
2) Manufacturer bazlı mode...
   Sonrası null: 0
3) Kalan null'lar 'unknown' yapılıyor...
   Final null: 0

Type doldurma tamamlandı!

Type dağılımı:
type
sedan          117634
SUV             90340
pickup          74027
truck           33492
other           21246
coupe           20649
hatchback       17236
van             10720
wagon           10532
convertible      7152
mini-van         4682
offroad           570
bus               314
Name: count, dtype: int64
```

Train Test Split Oranı : %80 Train %20 Test

Kullanılan Performans metrikleri : R2 , MAE , RMSE

Yapılan deneyler sonucunda, GPU desteğiyle hızlı eğitilebilmesi ve test verisi üzerinde en iyi performans metriklerini sağlaması nedeniyle XGBoost en başarılı model olarak seçilmiştir. Hiperparametre optimizasyonunda, GridSearchCV'ye kıyasla çok daha hızlı çalışması ve benzer verimlilikte sonuçlar üretmesi nedeniyle ağırlıklı olarak

RandomizedSearchCV kullanılmıştır. Optimizasyon sürecinde ağaç sayısı, maksimum derinlik, öğrenme oranı ve örnekleme oranı gibi temel hiperparametreler ayarlanmıştır. Model; araç yılı, kilometre, motor ve yakıt özellikleri, şanzıman, araç tipi, çekiş türü, üretici bilgisi ve coğrafi konum gibi sayısal ve kategorik özellikler kullanılarak eğitilmiştir.

Ekip Üyesi	Kullanılan Model	Hyperparameter Tuning Yöntemi	En Optimum Parametreler	Kullanılan Performans Metrikleri
Buğra	Random Forest Regressor	RandomizedSearchCV (n_iter=12, cv=3)	n_estimators=400, max_depth=20, min_samples_split=5, min_samples_leaf=2, max_features="sqrt"	RMSE, MAE, R²
Buğra	PyTorch Neural Network (MLP)	Manual Tuning	learning_rate=0.001, batch_size=32, epochs=50	RMSE, Loss, R²
Emirhan	XGBoost Regressor	RandomizedSearchCV (n_iter=30, cv=3)	n_estimators=1000, max_depth=8, learning_rate=0.05, subsample=0.8	RMSE, MAE, R²
Emirhan	LightGBM Regressor	RandomizedSearchCV (n_iter=30, cv=3)	n_estimators=500, max_depth=10, learning_rate=0.1, num_leaves=31	RMSE, MAE, R²
Utku	Bagging Ensemble (KNN)	RandomizedSearchCV (n_iter=10, cv=3)	n_estimators=200, max_samples=1.0, learning_rate=0.1	RMSE, MAE, R²
Utku	CatBoost	Randomized Search	n_estimators=800, max_depth=8, learning_rate=0.05, loss_function="RMSE"	RMSE, MAE, R²
Alper	HistGradientBoostingRegressor	Manual Tuning (iterative parameter refinement)	max_depth=15, learning_rate=0.03, max_iter=800	RMSE, MAE, R²
Alper	Stacking Regressor	Grid Search (meta-model & base model selection)	base_models=[RF, LGBM, XGB], final_estimator=LinearRegression	RMSE, MAE, R²

R² Score (↑)	RMSE ($) (↓)	MAE ($) (↓)
XGBoost : 0.8869	**Random Forest : 3,950.88**	**Bagging KNN : 2,033.50**
Random Forest : 0.8831	XGBoost : 3,992.31	Random Forest : 2,220.24
LightGBM : 0.8698	LightGBM : 4,389.76	XGBoost : 2,393.11
Bagging KNN : 0.8380	Bagging KNN : 5,153.38	Stacking Regressor : 2,426.28
Stacking Regressor : 0.8202	Stacking Regressor : 5,375.21	LightGBM : 2,745.36
CatBoost : 0.8178	CatBoost : 5,473.67	CatBoost : 2,660.70
HistGradientBoosting : 0.7299	HistGradientBoosting : 7,602.93	HistGradientBoosting : 4,152.12
PyTorch ResNet MLP : 0.7152	PyTorch ResNet MLP : 7,956.78	PyTorch ResNet MLP : 4,423.55