PYTHON İLE İKİNCİ EL OTO VERİSİ REGRESYON ANALİZİ

Mert AÇIKGÖZ
Fatih AKTAŞ
İsa ERGÜN
Mesut HARDAL
Emre MENOKAN
Oğuzhan TANRIKULU

Veri Odaklı Programlama Dersi – İş Analitiği YL.
İstanbul Şehir Üniversitesi - 2019

İÇİNDEKİLER

1. Projenin Amacı 2

2. CRISP-DM metodolojisi 2

İşi anlama 2

Veriyi Anlama 3

Veriyi Hazırlama 3

Modelleme 3

Değerlendirme 3

Dağıtım, Konumlandırma 3

3. Business Understanding 5

4. Data Understanding 6

5. Veri önişleme 10

6. Modelling 17

7. Evaluating 18

1. Projenin Amacı

www.kaggle.com dan alınan ikinci el oto verisinin python programlama dili kullanılarak fiyatlarının tahminini yapmaktır. Tahminler regresyon yöntemleri kullanılarak yapılmıştır. Proje tüm dünyada kabul görmüş CRISP-DM metodolojisi kullanılarak gerçekleştirilmiştir. Kullanılan programlama dili Python'dır. Proje kapsamında Python'ın Pandas, numpy, matplotlib, seaborn, math ve scikit-learn gibi hazır kütüphanelerinden faydalanılmıştır.

2. CRISP-DM metodolojisi

Veri madenciliği birçok farklı aşamadan oluşur. Neredeyse her veri madenciliği çalışmasında verilerin kaynaklardan toplanması ve entegrasyonu, verilerin temizlenmesi, modelin oluşturulması, modelin denenmesi ve sonuçların sunuma hazırlanması adımları karşımıza çıkar.

CRISP-DM (Cross Industry Standard Process Model for Data Mining) endüstri ve kullanılan yazılımdan bağımsız bir veri madenciliği süreç modelidir. CRISP-DM veri madenciliği süreçlerini, kullanılan yazılımdan ve endüstriden bağımsız standartlaştırmayı amaçlar.

CRISP-DM aşağıdaki fazlardan oluşur:

- **İşi anlama**

Bu adımda projenin hedeflerini ve ihtiyaçlarını iş perspektifinden anlamak, bu bilgiyi veri madenciliği sürecinde problem tanımı ve iş ön planı oluşturmak için kullanmak yer alır. Hedeflere şirket açısından bakıp, iş verenin varmak istediği noktayı anlamak ve bu problemi veri madenciliği problemine indirgemek önemlidir. Bunlar yapılırken sahip olunan verileri, yazılımları, donanımları ve proje sırasında ortaya çıkabilecek olası problemleri belirlemek gerekmektedir.

- **Veriyi Anlama**

Veri toplamak, veriyi tanımak, veri kalitesini belirleyip sorunlarını tanımlamak, gizli bilgi için hipotez oluşturmak gibi adımlardan oluşur.

- **Veriyi Hazırlama**

Verinin işlenmesi, eksik verilerin tahmini, verinin yapılandırılması, entegrasyonu, modellemelere uygun şekillendirilmesi adımlarından oluşur. Başlangıç veri kümesinden itibaren son veri kümesini oluşturmak için yapılan tüm aşamaları barındırır.

- **Modelleme**

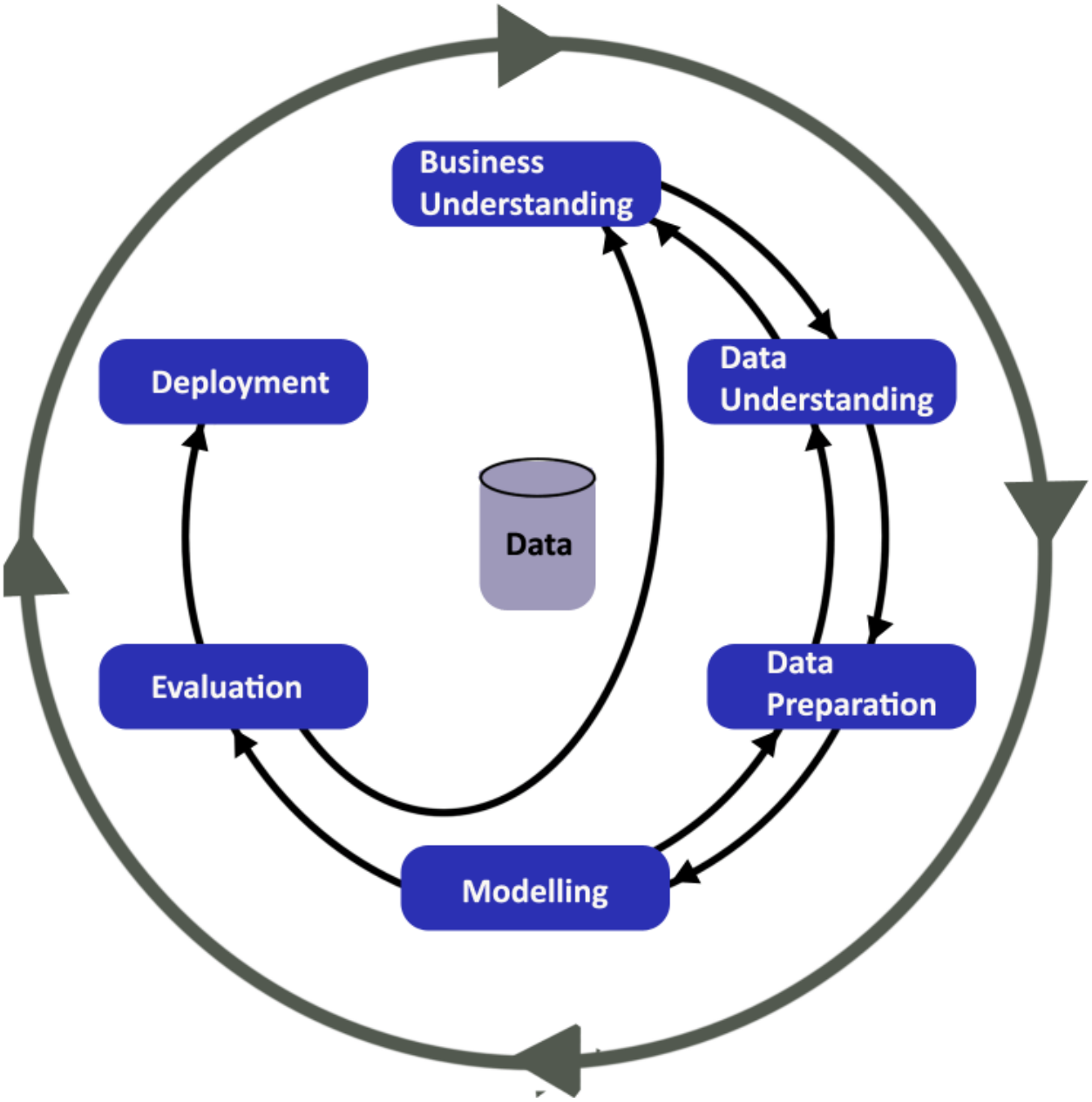
Modelleme tekniği seçilir ve yazılıma bağlı olarak hangi algoritmanın kullanılacağı da belirlenir. Nöral ağlar gibi bazı teknikler veri yapısına ilişkin özel gereksinimlere sahip olduğundan veri hazırlama aşaması burada tekrar döngüye girebilir. Modellerin parametrelerinin ve testlerinin belirlenmesi bu aşamada yer alır.

- **Değerlendirme**

Sonuçların test değerlerine göre teknik açıdan, proje aşamalarına göre de iş açısından iki farklı yönden değerlendirilmesini kapsar. Kayıp fonksiyonlara bağlı olarak bir veya daha fazla model oluşturulduğunda bunların yüksek kalite açısından değerlendirilebilmesi gerekir. Bunun için test sonuçlarına bakılarak en verimli modelin seçilmesi bu aşamaya girmektedir.

- **Dağıtım, Konumlandırma**

İş sorunlarının çözümü amacıyla modelin bir kod temsilini, işletim sistemine dağıtma aşamasıdır. En önemlisi bu kod temsili, modellemeye giden tüm veri adımlarını da içermelidir. Böylece model ham verileri model geliştirme sırasında olduğu gibi ele alacaktır. Sistemin sağlıklı çalışması için bakım planlaması ve nihai raporun sunulması ile proje sonlandırılır.


Business Understanding
Data Understanding
Data Preparation
Modelling
Evaluation
Deployment
Data

3. Business Understanding

Kaggle’dan alınan CraiglistVehicles verisinde araçların fiyatları, silindir sayıları, motor tipleri, yakıt sistemleri, renkleri gibi çeşitli kolonlar mevcut. Bu kolonların detaylarından “Data Understanding” bölümünde bahsedilecektir. Bu kolonlardan fiyat kolonu bizim bağlı değişkenimiz, diğer kolonlar da bağımsız değişkenlerimizdir. Eğer amacımızı detaylandırmamız gerekirse, bağımsız değişkenleri kullanılarak bağlı değişkenimizi tahmin etmeye yarayacak regresyon modelleri kurmak ve bu modelleri karşılaştırmaktır.

Craigslist, dünyanın en büyük ikinci el satılık araç koleksiyonudur, ancak hepsini aynı yerde toplamak çok zordur. Kaggle’da bu veri daha sonra üzerine ABD'de Craigslist'te kullanılan her araç girişini içeren bu veri kümesini oluşturmak için genişletildi.

4. Data Understanding

Önce kullandığımız veri bir dataframe'e yüklendi ve verinin formatı, kaç satır ve sütundan oluştuğu, kolonların veri tipleri gibi özellikleri aşağıdaki gibi kontrol edildi:

```
#kullanilacak verinin dataframe'e import islemi
df_ham = pd.read_csv('C:\\Users\\oguzh\\Desktop\\craigslistVehicles.csv', engine='python'
df = df_ham
```

```
#ilk on satırın gosterilmesi
df.head(10)
```

	url	city	city_url	price	year	manufacturer	make	condition	cylinders	fuel	...	transmis
0	https://abilene.craigslist.org/cto/d/aspermont...	abilene, TX	https://abilene.craigslist.org	9000	2009.0	chevrolet	suburban lt2	good	8 cylinders	gas	...	autor
1	https://abilene.craigslist.org/ctd/d/liberty-h...	abilene, TX	https://abilene.craigslist.org	31999	2012.0	ram	2500	NaN	NaN	diesel	...	autor
2	https://abilene.craigslist.org/ctd/d/liberty-h...	abilene, TX	https://abilene.craigslist.org	16990	2003.0	ram	3500	NaN	NaN	diesel	...	ma
3	https://abilene.craigslist.org/cto/d/merkel-20...	abilene, TX	https://abilene.craigslist.org	6000	2002.0	gmc	sierra 1500	good	8 cylinders	gas	...	autor
4	https://abilene.craigslist.org/cto/d/breckenri...	abilene, TX	https://abilene.craigslist.org	37000	2012.0	chevrolet	3500	excellent	8 cylinders	diesel	...	autor
5	https://abilene.craigslist.org/cto/d/abilene-t...	abilene, TX	https://abilene.craigslist.org	3700	2003.0	NaN	F150	fair	8 cylinders	gas	...	autor
6	https://abilene.craigslist.org/ctd/d/arlington...	abilene, TX	https://abilene.craigslist.org	19950	2013.0	ford	f-250	NaN	8 cylinders	gas	...	autor
7	https://abilene.craigslist.org/ctd/d/leander-2...	abilene, TX	https://abilene.craigslist.org	19999	2006.0	ram	2500	NaN	NaN	diesel	...	ma
8	https://abilene.craigslist.org/ctd/d/arlington...	abilene, TX	https://abilene.craigslist.org	33950	2015.0	ford	f-350	NaN	8 cylinders	diesel	...	autor
9	https://abilene.craigslist.org/ctd/d/arlington...	abilene, TX	https://abilene.craigslist.org	25950	2015.0	ford	f-350	NaN	8 cylinders	gas	...	autor

10 rows × 22 columns

```
#satir ve sutun gosterimi
df.shape
```

```
(550313, 22)
```

```
#kolonların gosterilmesi
df.columns
```

```
Index(['url', 'city', 'city_url', 'price', 'year', 'manufacturer', 'make',
       'condition', 'cylinders', 'fuel', 'odometer', 'title_status',
       'transmission', 'VIN', 'drive', 'size', 'type', 'paint_color',
       'image_url', 'desc', 'lat', 'long'],
      dtype='object')
```

Kolonların unique değerleri aşağıdaki kod ile görüntülendi.

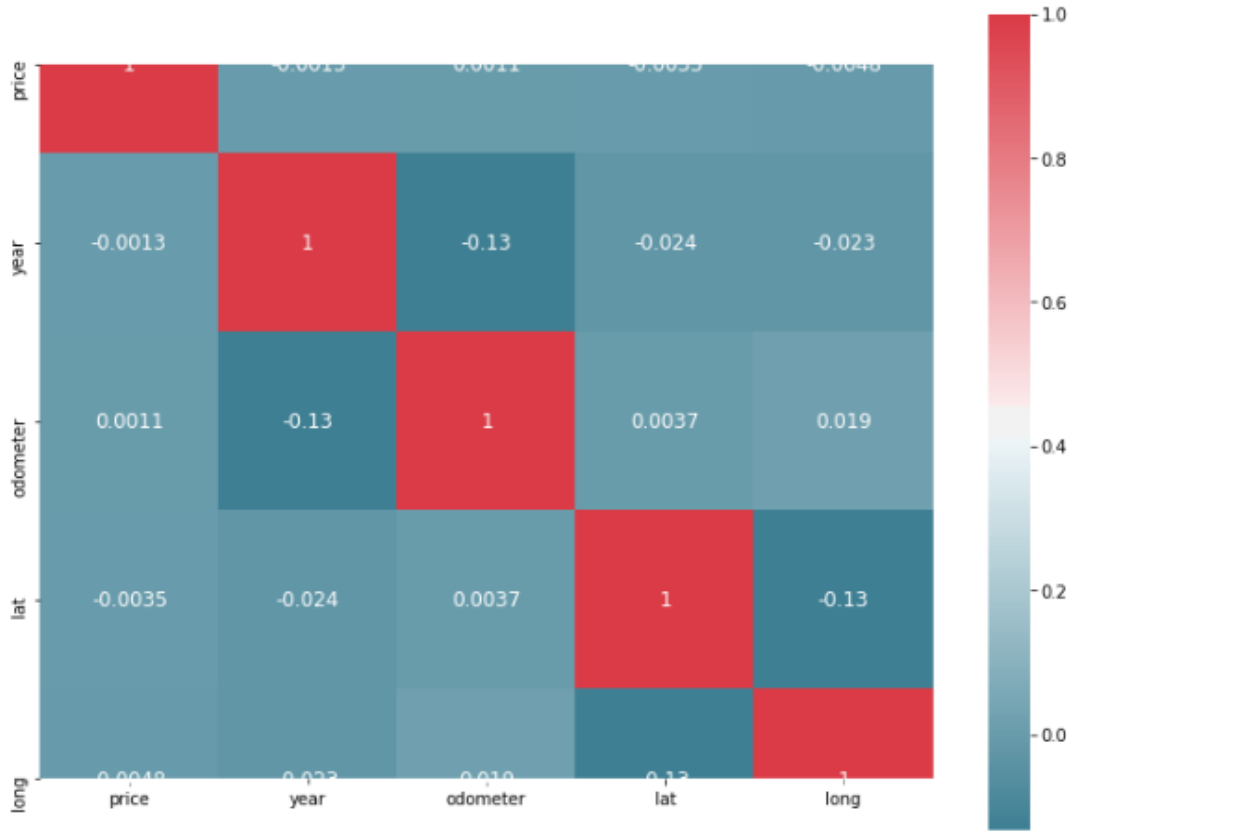
```
for i in range(len(df.columns)):
    print(df.iloc[:,i].unique()) # kolon distinct değerlerinin gosterimi
```

```
#data tiplerinin gosterimi
df.dtypes
```

```
url             object
city            object
city_url        object
price            int64
year           float64
manufacturer    object
make            object
condition       object
cylinders       object
fuel            object
odometer       float64
title_status    object
transmission    object
VIN             object
drive           object
size            object
type            object
paint_color     object
image_url       object
desc            object
lat            float64
long           float64
dtype: object
```

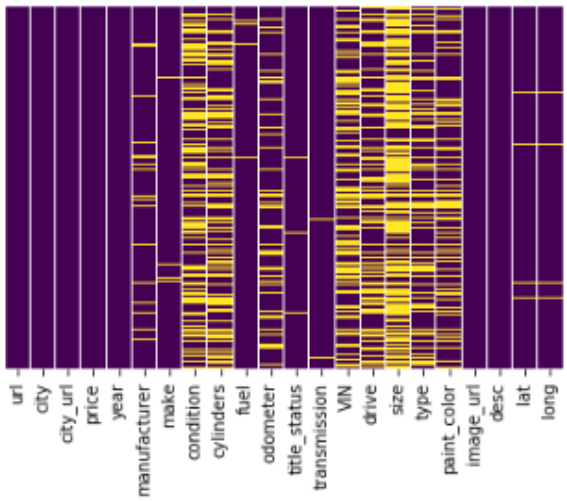
Numerik verilerin aralarındaki korelasyonu aşağıdaki kod ile görüntüledik. Verimiz henüz ön işlemlere tabi tutulmadığı ve tip dönüşümleri yapılmadığı için elde edilen korelasyon çıktısı anlamlı görünmüyor.

```
# numerik sutunların korelasyonlarinin plot ile gosteren fonksiyon
def plot_correlation_map( df ):
    corr = df.corr()
    _ , ax = plt.subplots( figsize =( 12 , 10 ) )
    cmap = sns.diverging_palette( 220 , 10 , as_cmap = True )
    _ = sns.heatmap(
        corr,
        cmap = cmap,
        square=True,
        cbar_kws={ 'shrink' : .9 },
        ax=ax,
        annot = True,
        annot_kws = { 'fontsize' : 12 }
    )
plot_correlation_map(df)
```



```
#eksik degerlerin grafiksel gosterimi
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis') # missing value check
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2846d178308>
```




```
#kolonlardaki null değerlerin sayısal olarak gösterilmesi
for i in range(len(df.columns)):
    x = df.columns[i]
    print(x, '-', df[df[x].isnull() == True].shape)
```

```
url - (0, 22)
city - (0, 22)
city_url - (0, 22)
price - (0, 22)
year - (1487, 22)
manufacturer - (26915, 22)
make - (9677, 22)
condition - (250074, 22)
cylinders - (218997, 22)
fuel - (4741, 22)
odometer - (110800, 22)
title_status - (4024, 22)
transmission - (4055, 22)
VIN - (239238, 22)
drive - (165838, 22)
size - (366256, 22)
type - (159179, 22)
paint_color - (180021, 22)
image_url - (26, 22)
desc - (30, 22)
lat - (11790, 22)
long - (11790, 22)
```

5. Veri önişleme

Kullanıln CRISP-DM metodolojisinin üçüncü adımına göre veri tip dönüşümleri, eksik değerlerin tamamlanması, encoding işlemleri, hangi kolonların modele dahil edilip edilmeyeceğinin belirlenmesi ve outlier değerlerin temizlenmesi gibi bazı önişleme adımlarının uygulanması gerekmektedir.

```
#belirli kolonların silinmesi
df.drop(columns = ['url', 'city_url', 'image_url','size', 'VIN', 'lat', 'long'], axis=1,inplace=True)
```

'Manufacturer' değerleri eksik olan satırların 'make' kolonundan elde edilerek doldurulması.

```
df_make_manufacturer_dropna = df.dropna(subset = ["make", "manufacturer"])
#dictionary olusturmadan önce missing value'ları drop ediyoruz, dictionary'nin hatalı olusaması için
dict_make_manufacturer = pd.Series(df_make_manufacturer_dropna.manufacturer.values,index=df_make_manufacturer_dropna.make).to_di
#make ve manufacturer için dictionary oluşturuyoruz.
df.manufacturer = df.manufacturer.fillna(df.make.map(dict_make_manufacturer))
#dictionary ile dataframe'in make alanlarını map edip, dictionary'nin manufacturer alanı ile df'in manufacturer = nan olanları
#güncelliyoruz
df[df.manufacturer.isnull() == True].shape #26915 - 25585 = 1330 tane missing value doldurduk
```

```
(25585, 15)
```

'Drive' değerleri eksik olan satırların 'make' kolonundan elde edilerek doldurulması.

```
df_make_drive_dropna = df.dropna(subset = ["make", "drive"])
#dictionary olusturmadan önce missing value'ları drop ediyoruz, dictionary'nin hatalı olusaması için
dict_make_drive = pd.Series(df_make_drive_dropna.type.values,index=df_make_drive_dropna.make).to_dict()
#make ve drive için dictionary oluşturuyoruz.
df.drive = df.drive.fillna(df.make.map(dict_make_drive))
#dictionary ile dataframe'in make alanlarını map edip, dictionary'nin drive alanı ile df'in drive = nan olanları
#güncelliyoruz
df[df.drive.isnull() == True].shape # 165818 - 49519 = 116299 tane missing value doldurduk
```

```
(49519, 15)
```

‘type’ değerleri eksik olan satırların ‘make’ kolonundan elde edilerek doldurulması.

```
df_make_type_dropna = df.dropna(subset = ["make", "type"])
#dictionary olusturmadan önce missing value'ları drop ediyoruz, dictionary'nin hatalı olusaması için
dict_make_type = pd.Series(df_make_type_dropna.type.values,index=df_make_type_dropna.make).to_dict()
#make ve type için dictionary oluşturuyoruz.
df.type = df.type.fillna(df.make.map(dict_make_type))
#dictionary ile dataframe'in make alanlarını map edip, dictionary'nin type alanı ile df'in type = nan olanları
#güncelliyoruz
df[df.type.isnull() == True].shape # 159179 - 21821 =  137358 tane missing value doldurduk
```

```
(21821, 15)
```

Odometer(kilometre) değerleri eksik olan satırlardaki araçların model yıllarına ait ortalama kilometre değerleri hesaplanarak verilerin doldurulması.

```
#global olarak kullanıyoruz
dict_odometer_mean = {} # year ve mean dictionary değişkeni

# nan odometer değerleri için mean değerlerini hesaplayan fonksiyon
def calc_nanvalues_mean_func(x,y):
    global  dict_odometer_mean
    if (math.isnan(y) == True):
        if x not in dict_odometer_mean:
            val = df[(df['year'] == x)]['odometer'].mean()
            return dict_odometer_mean.update({x:val})
        else:
            return dict_odometer_mean.get(x)
    else:
        return y
```

```
import math
```

```
# calc_nanvalues_mean_func fonksiyonunun her bir satır için çalıştırılması
df['odometer'] =  df.apply(lambda x: calc_nanvalues_mean_func(x['year'],  x['odometer']), axis=1)
```

‘price’ kolonundaki outlierlarların box plot quantile değerleri baz alınarak silinmesi

```
y = df['price']
removed_outliers = y.between(y.quantile(.10), y.quantile(.90))
```

```
df = df[removed_outliers]
```

```
df.shape
```

```
(440305, 15)
```

```
#1989 öncesi ve == 2020 için  veri az olduğu için alınmıyor..
df = df[(df['year'] > 1989) & (df['year'] < 2020)] #??? kayıt alındı.
```

Tüm kolonların unique değerlerin gösterilmesi

```
for i in range(len(df.columns)):
    print(df.iloc[:,i].value_counts()) # kolonların dağılımı
```

Kolonlarda doldurulabilecek eksik verilerin diğer kolonlardan bulunarak tamamlanmalarının sağlanmasından sonra herhangi bir eksik veri doldurma işlemi yapılamadığından, kalan eksik verili satırlar veriden hariç tutulmuştur.

```
#kalan eksik verilerin silinmesi islemi
df = df.dropna()
df.columns
```

```
Index(['city', 'price', 'year', 'manufacturer', 'make', 'condition',
       'cylinders', 'fuel', 'odometer', 'title_status', 'transmission',
       'drive', 'type', 'paint_color', 'desc'],
      dtype='object')
```

```
df.shape
```

```
(164162, 15)
```

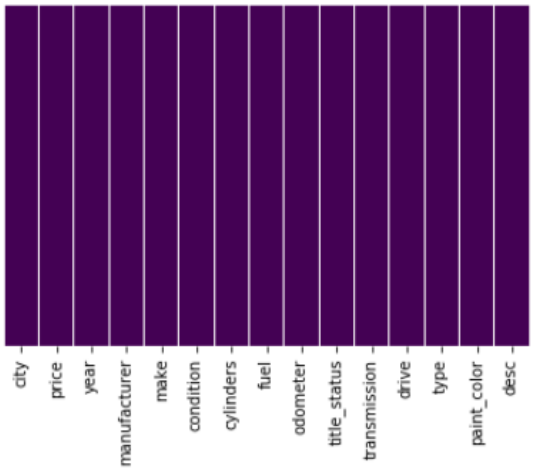
Eksik veriler çıkartıldıktan sonra ~ 164k satır kalmıştır.

```python
#float verilerin int donusumu
df['year'] = df['year'].astype(np.int64)
df['odometer'] = df['odometer'].astype(np.int64)
df.dtypes
```

```
city             object
price             int64
year              int64
manufacturer     object
make             object
condition        object
cylinders        object
fuel             object
odometer          int64
title_status     object
transmission     object
drive            object
type             object
paint_color      object
desc             object
dtype: object
```

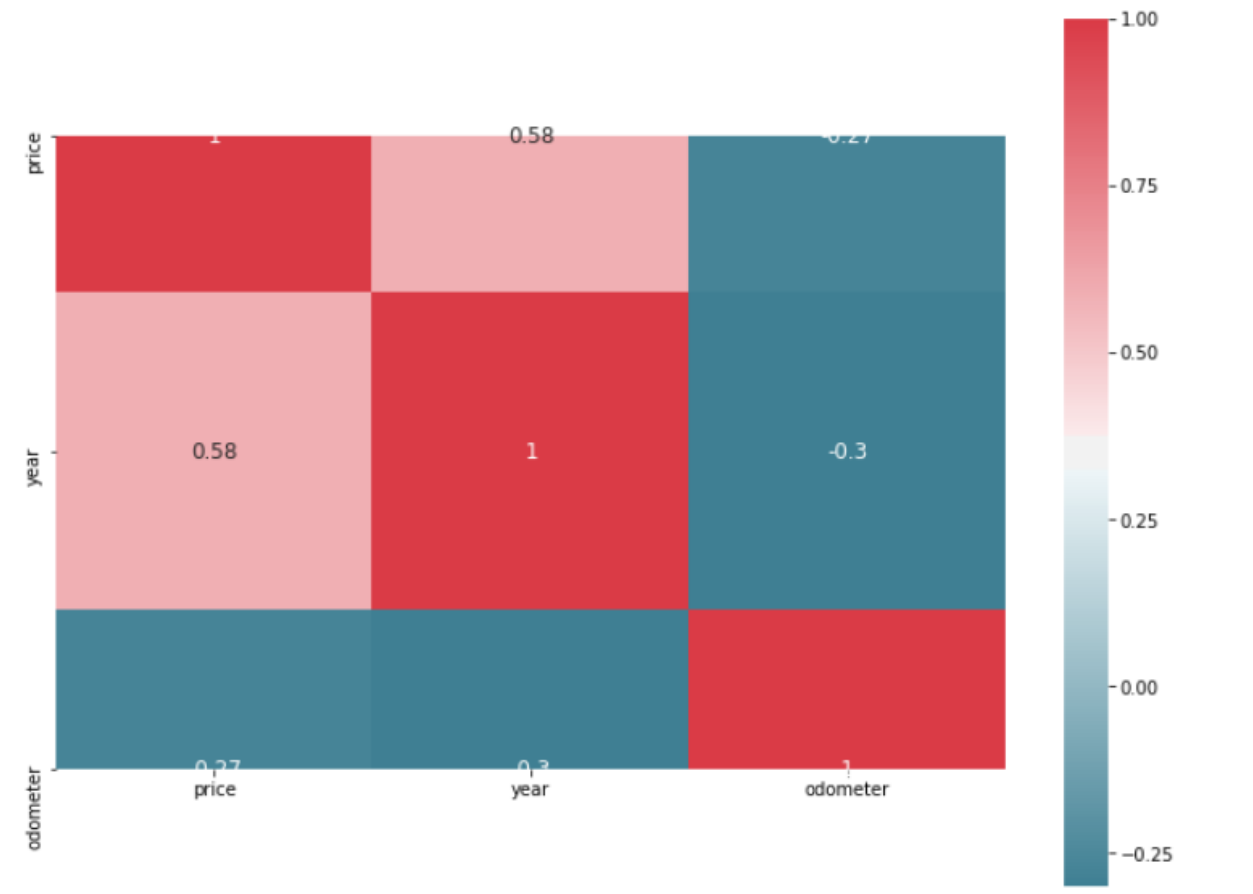
```python
sns.heatmap(df.isnull(),yticklabels=False,cbar=False,cmap='viridis') # OK!!!, eksik veri kalmadi.
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x2845de5e988>
```



Temizleme sonrası korelasyon görseli anlamlı hale geldi.

```
plot_correlation_map(df)
```



Regresyona verilecek kolonlardaki değerlerin anlamlı ve numerik hale getirilebilmesi için bu kolonlardaki değerlere one hot encoding işleminin uygulanması :

```
drive = pd.get_dummies(df['drive'], prefix = 'drive')
df.drop('drive', axis=1,inplace=True)
df = pd.concat([df, drive],axis=1)
```

```
type = pd.get_dummies(df['type'], prefix = 'type')
df.drop('type', axis=1,inplace=True)
df = pd.concat([df, type],axis=1)
```

```
transmission = pd.get_dummies(df['transmission'], prefix = 'transmission')
df.drop('transmission', axis=1,inplace=True)
df = pd.concat([df, transmission],axis=1)
```

```
paint_color = pd.get_dummies(df['paint_color'], prefix = 'paint_color')
df.drop('paint_color', axis=1,inplace=True)
df = pd.concat([df, paint_color],axis=1)
```

```
title_status = pd.get_dummies(df['title_status'], prefix = 'title_status')
df.drop('title_status', axis=1,inplace=True)
df = pd.concat([df, title_status],axis=1)
```

```
manufacturer = pd.get_dummies(df['manufacturer'], prefix = 'manufacturer')
df.drop('manufacturer', axis=1,inplace=True)
df = pd.concat([df, manufacturer],axis=1)
```

```
cylinders = pd.get_dummies(df['cylinders'], prefix = 'cylinders')
df.drop('cylinders', axis=1,inplace=True)
df = pd.concat([df, cylinders],axis=1)
```

```
condition = pd.get_dummies(df['condition'], prefix = 'condition')
df.drop('condition', axis=1,inplace=True)
df = pd.concat([df, condition],axis=1)
```

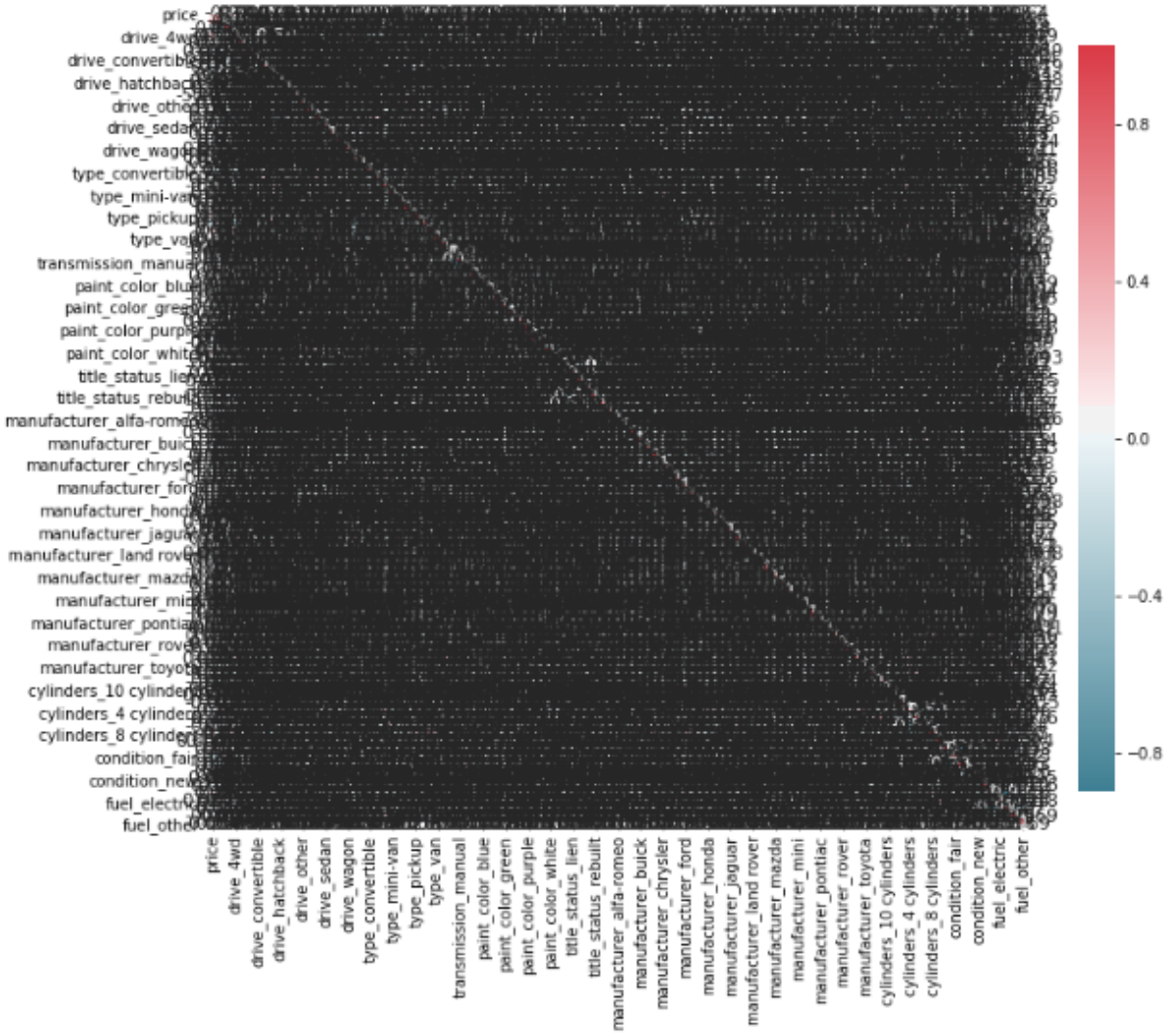
```
fuel = pd.get_dummies(df['fuel'], prefix = 'fuel')
df.drop('fuel', axis=1,inplace=True)
df = pd.concat([df, fuel],axis=1)
```

Encoding sonrası satır ve sütun sayısı gösterimi :

```
df.shape
```

```
(164162, 112)
```

Encoding sonrası korelasyon haritası:



Imputation için kullanıldıktan sonra make ve desc kolonlarının hariç bırakılması:

```
df.drop('make', axis=1,inplace=True)
df.drop('desc', axis=1,inplace=True)
```

6. Modelling

İkinci el araç verisinin fiyat tahminlemesi için lineer regresyon ve gradient boosted regressior modelleri kullanıldı.

```
#Lineer regresyon importu
from sklearn.linear_model import LinearRegression
```

```
#model değişkeni oluşturma
reg = LinearRegression()
```

```
#Bağımlı ve bağımsız değişkenlerin belirtilmesi
labels = df['price']
train1 = df.drop(["city",'price'],axis=1)
```

```
from sklearn.model_selection import train_test_split
```

```
x_train , x_test , y_train , y_test = train_test_split(train1 , labels , test_si
```

```
#from sklearn.preprocessing import StandardScaler
#sc = StandardScaler()
#X_train = sc.fit_transform(x_train)
#X_test  = sc.fit_transform(x_test)
#Y_train = sc.fit_transform(np.array(y_train).reshape(-1,1))
#Y_test  = sc.fit_transform(np.array(y_test).reshape(-1,1))
```

```
#Lineer Regresyon Modelinin eğitilmesi
reg.fit(x_train,y_train)
```

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None, normalize=False)
```

```
#Gradient Boosting Regressor Modelinin Kullanımı
from sklearn import ensemble
clf = ensemble.GradientBoostingRegressor(n_estimators = 400, max_depth = 5, min_samples_split = 2,
          learning_rate = 0.1, loss = 'ls')
```

```
#Gradient Boosting Regressor Modelinin eğitilmesi
clf.fit(x_train, y_train)
```

```
GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
                          learning_rate=0.1, loss='ls', max_depth=5,
                          max_features=None, max_leaf_nodes=None,
                          min_impurity_decrease=0.0, min_impurity_split=None,
                          min_samples_leaf=1, min_samples_split=2,
                          min_weight_fraction_leaf=0.0, n_estimators=400,
                          n_iter_no_change=None, presort='auto',
                          random_state=None, subsample=1.0, tol=0.0001,
                          validation_fraction=0.1, verbose=0, warm_start=False)
```

7. Evaluating

```
#Lineer Gegresyon modeli test verisinin tahmin edilmesi
tahmin = reg.predict(x_test)
print(tahmin)
```

```
[15463.58984357 13864.08297957 12508.04861084 ...  3316.7843446
  6813.50166782 11952.0149243 ]
```

```
from sklearn.metrics import mean_squared_error

# predicting on training data-set
y_train_predicted = reg.predict(x_train)

# predicting on test data-set
y_test_predict = tahmin

# evaluating the model on training dataset
rmse_train = np.sqrt(mean_squared_error(y_train, y_train_predicted))
r2_train = r2_score(y_train, y_train_predicted)

# evaluating the model on test dataset
rmse_test = np.sqrt(mean_squared_error(y_test, y_test_predict))
r2_test = r2_score(y_test, y_test_predict)

print("The Linear Regression model performance for the training set")
print("-------------------------------------------")
print("RMSE of training set is {}".format(rmse_train))
print("R2 score of training set is {}".format(r2_train))

print("\n")

print("The Linear Regression model performance for the test set")
print("-------------------------------------------")
print("RMSE of test set is {}".format(rmse_test))
print("R2 score of test set is {}".format(r2_test))
```

```
The Linear Regression model performance for the training set
-------------------------------------------
RMSE of training set is 3984.847555732764
R2 score of training set is 0.6428248597864326


The Linear Regression model performance for the test set
-------------------------------------------
RMSE of test set is 4029.0961700614375
R2 score of test set is 0.636387704461137
```

```
#GradientBoostingRegressor modeli test verisinin tahmin edilmesi
tahmin = clf.predict(x_test)
print(tahmin)
```

```
[15426.30367888 17235.43503442  9395.77762747 ...  2642.73123443
  5019.13185984 13594.63265251]
```

```
from sklearn.metrics import mean_squared_error

# predicting on training data-set
y_train_predicted = clf.predict(x_train)

# predicting on test data-set
y_test_predict = tahmin

# evaluating the model on training dataset
rmse_train = np.sqrt(mean_squared_error(y_train, y_train_predicted))
r2_train = r2_score(y_train, y_train_predicted)

# evaluating the model on test dataset
rmse_test = np.sqrt(mean_squared_error(y_test, y_test_predict))
r2_test = r2_score(y_test, y_test_predict)

print("The Gradient Boosting Regressor model performance for the training set")
print("--------------------------------------------")
print("RMSE of training set is {}".format(rmse_train))
print("R2 score of training set is {}".format(r2_train))

print("\n")

print("The Gradient Boosting Regressor model performance for the test set")
print("--------------------------------------------")
print("RMSE of test set is {}".format(rmse_test))
print("R2 score of test set is {}".format(r2_test))
```

```
The Gradient Boosting Regressor model performance for the training set
--------------------------------------------
RMSE of training set is 2676.293002578639
R2 score of training set is 0.8388890280427543


The Gradient Boosting Regressor model performance for the test set
--------------------------------------------
RMSE of test set is 2846.885450758004
R2 score of test set is 0.8184637838334818
```