CARVANA

In [4]:
```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import datetime

from sklearn.feature_selection import VarianceThreshold, RFECV
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import GridSearchCV, validation_curve

from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression

from sklearn.ensemble import VotingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier


import warnings
warnings.filterwarnings("ignore")

pd.options.display.max_rows = 50
pd.options.display.max_columns = 50
```

In [5]:
```python
data = pd.read_csv("carvana.csv", index_col=0 ,sep=',')
```

In [6]:
```python
data
```

Out[6]:

	PurchDate	Auction	VehYear	VehicleAge	Make	Model	Trim	SubModel	Color	Transmission	WheelTypeID	WheelTyp
RefId												
1	1/7/2009 0:12	ADESA	2005	4	DODGE	STRATUS V6	SXT	4D SEDAN SXT FFV	MAROON	AUTO	2.0	Cove
2	1/7/2009 0:12	ADESA	2004	5	DODGE	NEON	SXT	4D SEDAN	SILVER	AUTO	1.0	Allo
3	1/7/2009 0:12	ADESA	2005	4	FORD	FOCUS	ZX3	2D COUPE ZX3	SILVER	MANUAL	2.0	Cove
4	1/7/2009 0:12	ADESA	2004	5	MITSUBISHI	GALANT 4C	ES	4D SEDAN ES	WHITE	AUTO	2.0	Cove
5	1/21/2009 0:12	ADESA	2004	5	DODGE	1500 RAM PICKUP 2WD	ST	QUAD CAB 4.7L SLT	WHITE	AUTO	1.0	Allo
...	...	...	...	...	...	...	...	...	...	...	...	
6794	1/18/2009 0:11	ADESA	2006	3	SUZUKI	FORENZA	NaN	4D SEDAN	RED	AUTO	2.0	Cove
6795	1/24/2009 0:11	ADESA	2005	4	JEEP	LIBERTY 4WD V6	Spo	4D SUV	RED	AUTO	1.0	Allo
6796	1/24/2009 0:11	ADESA	2002	7	CHEVROLET	TRAILBLAZER EXT 2WD	LT	4D SUV 4.2L LT	BEIGE	AUTO	1.0	Allo
6797	1/24/2009 0:11	ADESA	2006	3	JEEP	COMMANDER 2WD V8	Bas	4D WAGON	BLACK	AUTO	2.0	Cove
6798	1/2/2009 0:12	ADESA	2002	7	DODGE	1500 RAM PICKUP 2WD	ST	QUAD CAB 4.7L SLT	GOLD	AUTO	1.0	Allo

6798 rows × 33 columns

In [7]:
```python
data.shape
```

Out[7]:
```
(6798, 33)
```

In [8]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6798 entries, 1 to 6798
Data columns (total 33 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   PurchDate                          6798 non-null   object
 1   Auction                            2983 non-null   object
 2   VehYear                            6798 non-null   int64
 3   VehicleAge                         6798 non-null   int64
 4   Make                               6798 non-null   object
 5   Model                              6798 non-null   object
 6   Trim                               6566 non-null   object
 7   SubModel                           6798 non-null   object
 8   Color                              6798 non-null   object
 9   Transmission                       6798 non-null   object
 10  WheelTypeID                        6515 non-null   float64
 11  WheelType                          6459 non-null   object
 12  VehOdo                             6798 non-null   int64
 13  Nationality                        6797 non-null   object
 14  Size                               6797 non-null   object
 15  TopThreeAmericanName               6797 non-null   object
 16  MMRAcquisitionAuctionAveragePrice  6797 non-null   float64
 17  MMRAcquisitionAuctionCleanPrice    6797 non-null   float64
 18  MMRAcquisitionRetailAveragePrice   6797 non-null   float64
 19  MMRAcquisitonRetailCleanPrice      6797 non-null   float64
 20  MMRCurrentAuctionAveragePrice      6777 non-null   float64
 21  MMRCurrentAuctionCleanPrice        6777 non-null   float64
 22  MMRCurrentRetailAveragePrice       6777 non-null   float64
 23  MMRCurrentRetailCleanPrice         6777 non-null   float64
 24  PRIMEUNIT                          339 non-null    object
 25  AUCGUART                           339 non-null    object
 26  BYRNO                              6798 non-null   int64
 27  VNZIP1                             6798 non-null   int64
 28  VNST                               1131 non-null   object
 29  VehBCost                           6798 non-null   float64
 30  IsOnlineSale                       6798 non-null   int64
 31  WarrantyCost                       6798 non-null   int64
 32  IsBadBuy                           6798 non-null   int64
dtypes: float64(10), int64(8), object(15)
memory usage: 1.8+ MB
```

In [9]:
```python
#Auction, Primeunit, Aucguart, VNST imaju previse nedostajucih vrednosti, pa ih iskljucujemo iz dalje analize
```

In [10]:
```python
data.drop(['Auction', 'PRIMEUNIT', 'AUCGUART', 'VNST'], axis=1, inplace=True)
```

In [11]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6798 entries, 1 to 6798
Data columns (total 29 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   PurchDate                          6798 non-null   object
 1   VehYear                            6798 non-null   int64
 2   VehicleAge                         6798 non-null   int64
 3   Make                               6798 non-null   object
 4   Model                              6798 non-null   object
 5   Trim                               6566 non-null   object
 6   SubModel                           6798 non-null   object
 7   Color                              6798 non-null   object
 8   Transmission                       6798 non-null   object
 9   WheelTypeID                        6515 non-null   float64
 10  WheelType                          6459 non-null   object
 11  VehOdo                             6798 non-null   int64
 12  Nationality                        6797 non-null   object
 13  Size                               6797 non-null   object
 14  TopThreeAmericanName               6797 non-null   object
 15  MMRAcquisitionAuctionAveragePrice  6797 non-null   float64
 16  MMRAcquisitionAuctionCleanPrice    6797 non-null   float64
 17  MMRAcquisitionRetailAveragePrice   6797 non-null   float64
 18  MMRAcquisitonRetailCleanPrice      6797 non-null   float64
 19  MMRCurrentAuctionAveragePrice      6777 non-null   float64
 20  MMRCurrentAuctionCleanPrice        6777 non-null   float64
 21  MMRCurrentRetailAveragePrice       6777 non-null   float64
 22  MMRCurrentRetailCleanPrice         6777 non-null   float64
 23  BYRNO                              6798 non-null   int64
 24  VNZIP1                             6798 non-null   int64
 25  VehBCost                           6798 non-null   float64
 26  IsOnlineSale                       6798 non-null   int64
 27  WarrantyCost                       6798 non-null   int64
 28  IsBadBuy                           6798 non-null   int64
dtypes: float64(10), int64(8), object(11)
memory usage: 1.6+ MB
```

Deskriptivna statistika

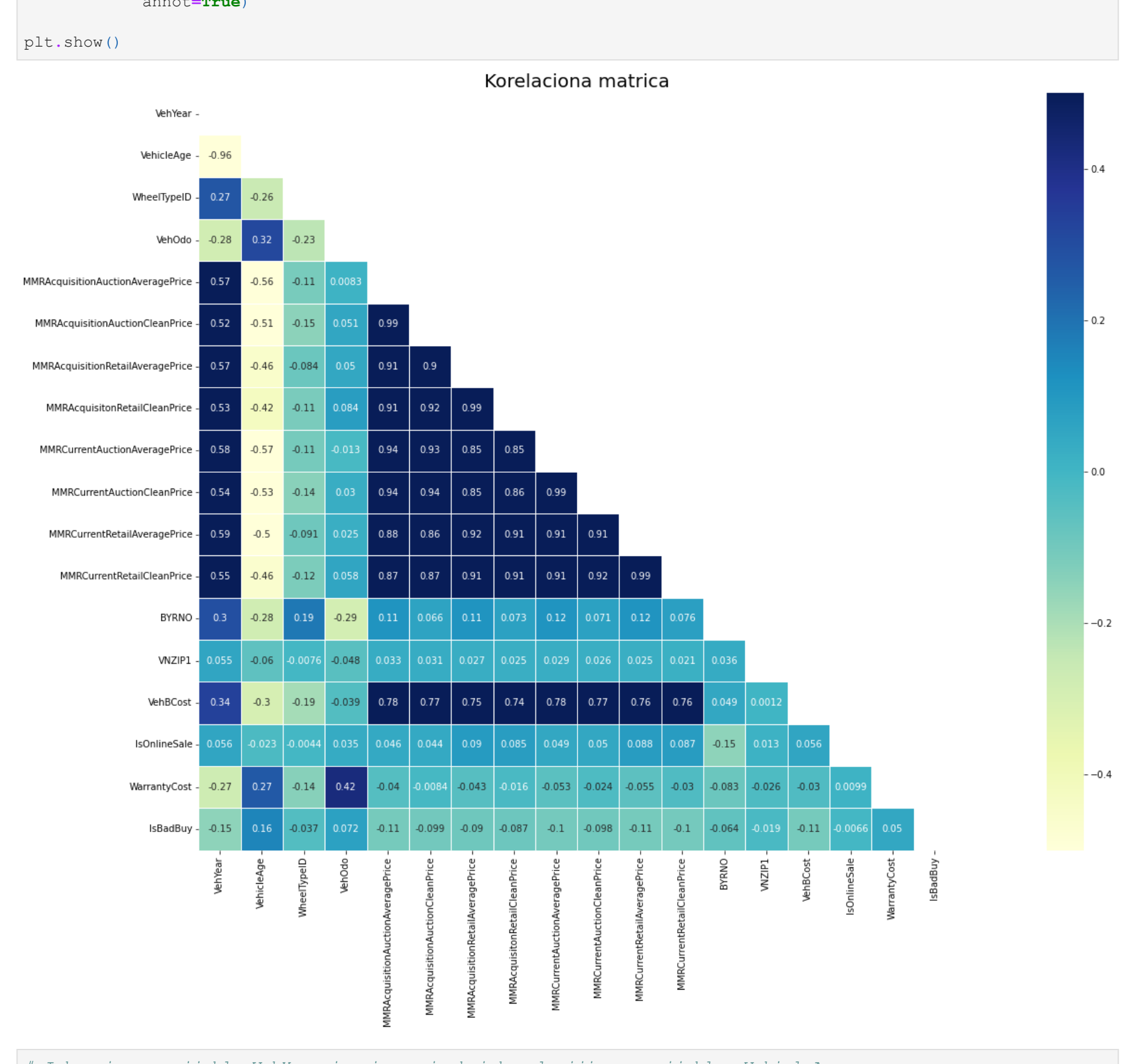
In [13]:
```python
pd.set_option('display.max_columns', 35)
data.describe(include = "all")
```

Out[13]:

	PurchDate	VehYear	VehicleAge	Make	Model	Trim	SubModel	Color	Transmission	WheelTypeID	WheelType	Ve
count	6798	6798.000000	6798.000000	6798	6798	6566	6798	6798	6798	6515.000000	6459	6798.0
unique	503	NaN	NaN	32	632	104	465	16	2	NaN	2	
top	1/30/2010 0:09	NaN	NaN	CHEVROLET	PT CRUISER	Bas	4D SEDAN	SILVER	AUTO	NaN	Alloy	
freq	42	NaN	NaN	1574	216	1316	1449	1379	6547	NaN	3376	
mean	NaN	2005.332451	4.183142	NaN	NaN	NaN	NaN	NaN	NaN	1.490407	NaN	71499.3
std	NaN	1.733395	1.719690	NaN	NaN	NaN	NaN	NaN	NaN	0.516856	NaN	14646.2
min	NaN	2001.000000	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	1.000000	NaN	15249.0
25%	NaN	2004.000000	3.000000	NaN	NaN	NaN	NaN	NaN	NaN	1.000000	NaN	62002.7
50%	NaN	2005.000000	4.000000	NaN	NaN	NaN	NaN	NaN	NaN	1.000000	NaN	73562.0
75%	NaN	2007.000000	5.000000	NaN	NaN	NaN	NaN	NaN	NaN	2.000000	NaN	82551.7
max	NaN	2009.000000	9.000000	NaN	NaN	NaN	NaN	NaN	NaN	3.000000	NaN	112029.0

Korelacija

In [15]:
```python
corr = data.select_dtypes(np.number).corr()
plt.figure(figsize=(35,15))
plt.title('Korelaciona matrica', fontsize=20)

sns.heatmap(corr,
           vmin=-.5, #sve preko 0.5 (ispod -0.5) ce biti predstavljeno kao srednja i visoka korelisanost
           vmax =.5,
           mask = np.triu(np.ones_like(corr, dtype=np.bool)),
           cmap = "YlGnBu",
           center=0,
           square=True,
           linewidths=.5,
           annot=True)

plt.show()
```



In [16]:
```python
# Izbacujemo varijablu VehYear jer je u visokoj korelaciji sa varijablom VehicleAge
# WheelTypeID i BYRNO izbacujemo jer nemaju nikakvog znacaja za nasu analizu
# VNZIP1 smatramo da je beznacajan, a i ima slabu korelaciju sa svim promenljivim
```

In [17]:
```python
data.drop(['VehYear', 'WheelTypeID', 'BYRNO', 'VNZIP1'], axis=1, inplace=True)
```

In [18]:
```python
data
```

Out[18]:

	PurchDate	VehicleAge	Make	Model	Trim	SubModel	Color	Transmission	WheelType	VehOdo	Nationality	Siz
RefId												
1	1/7/2009 0:12	4	DODGE	STRATUS V6	SXT	4D SEDAN SXT FFV	MAROON	AUTO	Covers	73807	AMERICAN	MEDIUM
2	1/7/2009 0:12	5	DODGE	NEON	SXT	4D SEDAN	SILVER	AUTO	Alloy	65617	AMERICAN	COMPAC
3	1/7/2009 0:12	4	FORD	FOCUS	ZX3	2D COUPE ZX3	SILVER	MANUAL	Covers	69367	AMERICAN	COMPAC
4	1/7/2009 0:12	5	MITSUBISHI	GALANT 4C	ES	4D SEDAN ES	WHITE	AUTO	Covers	81054	OTHER ASIAN	MEDIUM
5	1/21/2009 0:12	5	DODGE	1500 RAM PICKUP 2WD	ST	QUAD CAB 4.7L SLT	WHITE	AUTO	Alloy	76173	AMERICAN	LARG TRUC
...	...	...	...	...	...	...	...	...	...	...	...	
6794	1/18/2009 0:11	3	SUZUKI	FORENZA	NaN	4D SEDAN	RED	AUTO	Covers	60549	OTHER ASIAN	MEDIUM
6795	1/24/2009 0:11	4	JEEP	LIBERTY 4WD V6	Spo	4D SUV	RED	AUTO	Alloy	85377	AMERICAN	SMAL SU
6796	1/24/2009 0:11	7	CHEVROLET	TRAILBLAZER EXT 2WD	LT	4D SUV 4.2L LT	BEIGE	AUTO	Alloy	75700	AMERICAN	MEDIUM SU
6797	1/24/2009 0:11	3	JEEP	COMMANDER 2WD V8	Bas	4D WAGON	BLACK	AUTO	Covers	69941	AMERICAN	LARG SU
6798	1/2/2009 0:12	7	DODGE	1500 RAM PICKUP 2WD	ST	QUAD CAB 4.7L SLT	GOLD	AUTO	Alloy	93744	AMERICAN	LARG TRUC

6798 rows × 25 columns

Nedostajuce vrednosti

In [20]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6798 entries, 1 to 6798
Data columns (total 25 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   PurchDate                          6798 non-null   object
 1   VehicleAge                         6798 non-null   int64
 2   Make                               6798 non-null   object
 3   Model                              6798 non-null   object
 4   Trim                               6566 non-null   object
 5   SubModel                           6798 non-null   object
 6   Color                              6798 non-null   object
 7   Transmission                       6798 non-null   object
 8   WheelType                          6459 non-null   object
 9   VehOdo                             6798 non-null   int64
 10  Nationality                        6797 non-null   object
 11  Size                               6797 non-null   object
 12  TopThreeAmericanName               6797 non-null   object
 13  MMRAcquisitionAuctionAveragePrice  6797 non-null   float64
 14  MMRAcquisitionAuctionCleanPrice    6797 non-null   float64
 15  MMRAcquisitionRetailAveragePrice   6797 non-null   float64
 16  MMRAcquisitonRetailCleanPrice      6797 non-null   float64
 17  MMRCurrentAuctionAveragePrice      6777 non-null   float64
 18  MMRCurrentAuctionCleanPrice        6777 non-null   float64
 19  MMRCurrentRetailAveragePrice       6777 non-null   float64
 20  MMRCurrentRetailCleanPrice         6777 non-null   float64
 21  VehBCost                           6798 non-null   float64
 22  IsOnlineSale                       6798 non-null   int64
 23  WarrantyCost                       6798 non-null   int64
 24  IsBadBuy                           6798 non-null   int64
dtypes: float64(9), int64(5), object(11)
memory usage: 1.3+ MB
```

In [21]:
```python
# Trim ima 232 nedostajuce vrednosti
# WheelType ima 339 NA
# 10 - 16 jedna NA, proveri gde je i da li da se izbacuje ceo red
# 17 - 20 ima 21 NA
```

In [22]:
```python
data.isna().sum()
```

Out[22]:
```
PurchDate                              0
VehicleAge                             0
Make                                   0
Model                                  0
Trim                                 232
SubModel                               0
Color                                  0
Transmission                           0
WheelType                            339
VehOdo                                 0
Nationality                            1
Size                                   1
TopThreeAmericanName                   1
MMRAcquisitionAuctionAveragePrice      1
MMRAcquisitionAuctionCleanPrice        1
MMRAcquisitionRetailAveragePrice       1
MMRAcquisitonRetailCleanPrice          1
MMRCurrentAuctionAveragePrice         21
MMRCurrentAuctionCleanPrice           21
MMRCurrentRetailAveragePrice          21
MMRCurrentRetailCleanPrice            21
VehBCost                               0
IsOnlineSale                           0
WarrantyCost                           0
IsBadBuy                               0
dtype: int64
```

In [23]:
```python
data.loc[data['Nationality'].isna(),:]
# Proverile smo da li je mozda jedna NA za 6 varijabli u istom redu, posto smo videle da nije, nastavljamo dalj
```

Out[23]:

	PurchDate	VehicleAge	Make	Model	Trim	SubModel	Color	Transmission	WheelType	VehOdo	Nationality	Size	TopThreeAmerica
RefId													
6496	1/24/2010 0:02	2	JEEP	PATRIOT 2WD 4C	Spo	4D SUV SPORT	BLACK	AUTO	Covers	61596	NaN	NaN	

In [24]:
```python
# Trim
```

In [25]:
```python
data['Trim'].value_counts()
```

Out[25]:
```
Bas    1316
LS      921
SE      859
SXT     338
LT      318
       ...
RT        1
Max       1
Ove       1
ZTW       1
ZX2       1
Name: Trim, Length: 104, dtype: int64
```

In [26]:
```python
# Posto vidimo da imamo 104 razlicitih vrednosti i da dosta njih imaju frekvenciju 1, kako bismo sprecili
# pretreniranost modela, izbacicemo ovu varijablu
```

In [27]:
```python
data = data.drop('Trim',axis=1)
```

In [28]:
```python
# WheelType
```

In [29]:
```python
data['WheelType'].value_counts()
```

Out[29]:
```
Alloy     3376
Covers    3083
Name: WheelType, dtype: int64
```

In [30]:
```python
# Dominantna kategorija ove varijable je Alloy, tako da cemo nedostajuce vrednosti zameniti njom
```

In [31]:
```python
data.loc[data['WheelType'].isna(),'WheelType'] = 'Alloy'
```

In [32]:
```python
# 'Nationality' i 'Make'
```

In [33]:
```python
pd.crosstab(data['Make'], data['Nationality'])
```

Out[33]:

Nationality	AMERICAN	OTHER	OTHER ASIAN	TOP LINE ASIAN
Make				
ACURA	0	0	4	0
BUICK	63	0	0	0
CADILLAC	1	0	0	0
CHEVROLET	1574	0	0	0
CHRYSLER	827	0	0	0
DODGE	1187	0	0	0
FORD	1059	0	0	0
GMC	61	0	0	0
HONDA	0	0	0	52
HUMMER	1	0	0	0
HYUNDAI	0	0	181	0
INFINITI	0	0	6	0
ISUZU	0	0	8	0
JEEP	149	0	0	0
KIA	0	0	232	0
LEXUS	0	0	1	0
LINCOLN	7	0	0	0
MAZDA	0	0	112	0
MERCURY	81	0	0	0
MINI	0	5	0	0
MITSUBISHI	0	0	97	0
NISSAN	0	0	0	195
OLDSMOBILE	22	0	0	0
PLYMOUTH	1	0	0	0
PONTIAC	395	0	0	0
SATURN	211	0	0	0
SCION	0	0	5	0
SUBARU	0	0	1	0
SUZUKI	0	0	132	0
TOYOTA	0	0	0	109
VOLKSWAGEN	0	14	0	0
VOLVO	0	4	0	0

In [34]:
```python
# Zakljucujemo da su ove dve varijable visoko korelisane, a i da za Ameriku imamo mnogo vise podataka u odnosu
# pa cemo iz tog razloga izbaciti Nationality
```

In [35]:
```python
data = data.drop('Nationality',axis=1)
```

In [36]:
```python
# Size
```

In [37]:
```python
data['Size'].value_counts()
```

Out[37]:
```
MEDIUM         2888
LARGE           786
MEDIUM SUV      760
COMPACT         629
VAN             550
LARGE TRUCK     323
SMALL SUV       221
CROSSOVER       174
SPECIALTY       169
LARGE SUV       134
SMALL TRUCK      90
SPORTS           73
Name: Size, dtype: int64
```

In [38]:
```python
# Vidimo da je dominantna kategorija MEDIUM, pa cemo nedostajucu vrednost popuniti njom
```

In [39]:
```python
data.loc[data['Size'].isna(),'Size'] = 'MEDIUM'
```

In [40]:
```python
# TopThreeAmericanName
```

In [41]:
```python
data['TopThreeAmericanName'].value_counts()
```

Out[41]:
```
GM          2328
CHRYSLER    2164
OTHER       1158
FORD        1147
Name: TopThreeAmericanName, dtype: int64
```

In [42]:
```python
# Vidimo da je dominantna kategorija GM, pa cemo nedostajucu vrednost popuniti njom
```

In [43]:
```python
data.loc[data['TopThreeAmericanName'].isna(),'TopThreeAmericanName'] = 'GM'
```

In [44]:
```python
# Provericemo da li je nedostajuca vrednost isti red, pa ako jeste, izbacicemo tu opservaciju
```

In [45]:
```python
data.loc[data['MMRAcquisitionAuctionAveragePrice'].isna(),:]
```

Out[45]:

	PurchDate	VehicleAge	Make	Model	SubModel	Color	Transmission	WheelType	VehOdo	Size	TopThreeAmericanName	MMRA
RefId												
530	1/13/2010 0:07	6	NISSAN	TITAN PICKUP 2WD V8	EXT CAB 5.6L	WHITE	AUTO	Alloy	74705	LARGE TRUCK	OTHER	

In [46]:
```python
data = data.drop(530)
```

In [47]:
```python
data.iloc[525:535, :]
```

Out[47]:

	PurchDate	VehicleAge	Make	Model	SubModel	Color	Transmission	WheelType	VehOdo	Size	TopThreeAmericanNa
RefId											
526	1/12/2010 0:07	6	CHEVROLET	MALIBU V6	4D SEDAN	WHITE	AUTO	Covers	53861	MEDIUM	C
527	1/12/2010 0:07	4	DODGE	STRATUS V6	4D SEDAN SXT FFV	SILVER	AUTO	Covers	58329	MEDIUM	CHRYSL
528	1/12/2010 0:07	4	DODGE	STRATUS V6	4D SEDAN SXT FFV	SILVER	AUTO	Covers	48151	MEDIUM	CHRYSL
529	1/13/2010 0:07	6	DODGE	1500 RAM PICKUP 2WD	QUAD CAB 4.7L SLT	WHITE	AUTO	Alloy	77465	LARGE TRUCK	CHRYSL
531	1/23/2010 0:07	3	CHEVROLET	IMPALA	4D SEDAN LS	WHITE	AUTO	Covers	40651	LARGE	C
532	1/23/2010 0:07	4	DODGE	STRATUS V6	4D SEDAN SXT FFV	WHITE	AUTO	Covers	36198	MEDIUM	CHRYSL
533	1/19/2010 0:10	3	CHEVROLET	HHR	4D SUV 2.2L LS	GOLD	AUTO	Covers	67128	MEDIUM	C
534	1/19/2010 0:10	3	FORD	FIVE HUNDRED	4D SEDAN SEL	WHITE	AUTO	Alloy	73905	LARGE	FO
535	1/26/2010 0:10	4	DODGE	1500 RAM PICKUP 2WD	REG CAB 4.7L	WHITE	AUTO	Covers	82142	LARGE TRUCK	CHRYSL
536	1/26/2010 0:10	6	CHEVROLET	TRAILBLAZER 2WD 6C	4D SUV 4.2L LS	GREY	AUTO	Alloy	88479	MEDIUM SUV	C

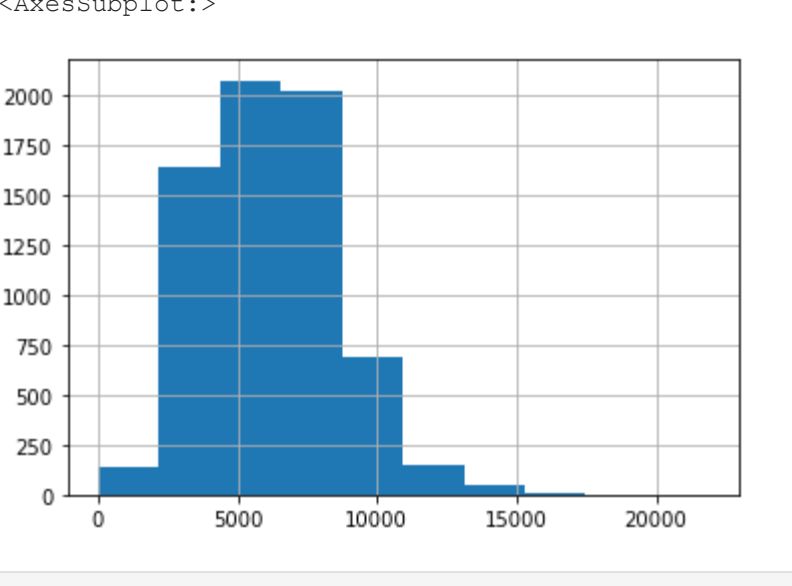
In [48]:
```python
data.isna().sum()
```

Out[48]:
```
PurchDate                             0
VehicleAge                            0
Make                                  0
Model                                 0
SubModel                              0
Color                                 0
Transmission                          0
WheelType                             0
VehOdo                                0
Size                                  0
TopThreeAmericanName                  0
MMRAcquisitionAuctionAveragePrice     0
MMRAcquisitionAuctionCleanPrice       0
MMRAcquisitionRetailAveragePrice      0
MMRAcquisitonRetailCleanPrice         0
MMRCurrentAuctionAveragePrice        20
MMRCurrentAuctionCleanPrice          20
MMRCurrentRetailAveragePrice         20
MMRCurrentRetailCleanPrice           20
VehBCost                              0
IsOnlineSale                          0
WarrantyCost                          0
IsBadBuy                              0
dtype: int64
```

In [49]:
```python
# Sredjivanje MMR varijabli
```

In [50]:
```python
data['MMRCurrentAuctionAveragePrice'].hist()
```

Out[50]:
```
<AxesSubplot:>
```



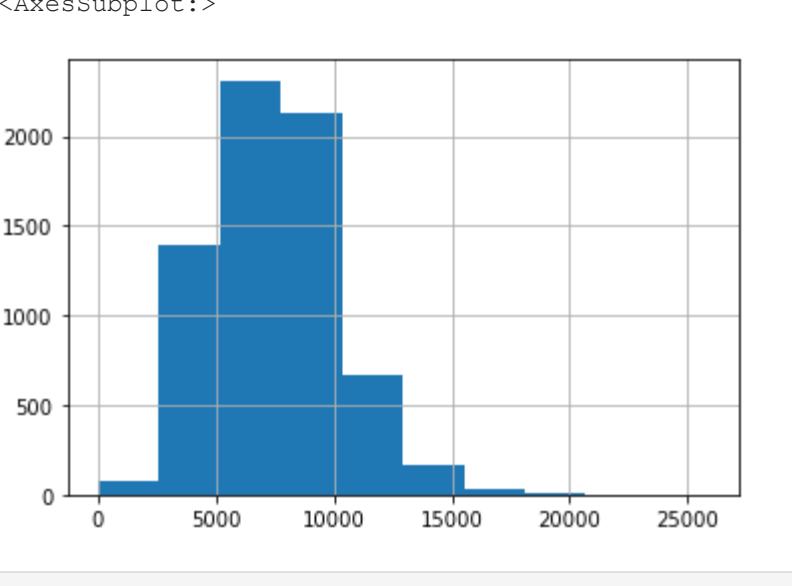
In [51]:
```python
import scipy
```

In [52]:
```python
scipy.stats.shapiro(data.loc[data['MMRCurrentAuctionAveragePrice'].isna()==False,'MMRCurrentAuctionAveragePrice
```

Out[52]:
```
ShapiroResult(statistic=0.9828617572784424, pvalue=7.471559996850242e-28)
```

In [53]:
```python
data['MMRCurrentAuctionCleanPrice'].hist()
```

Out[53]:
```
<AxesSubplot:>
```

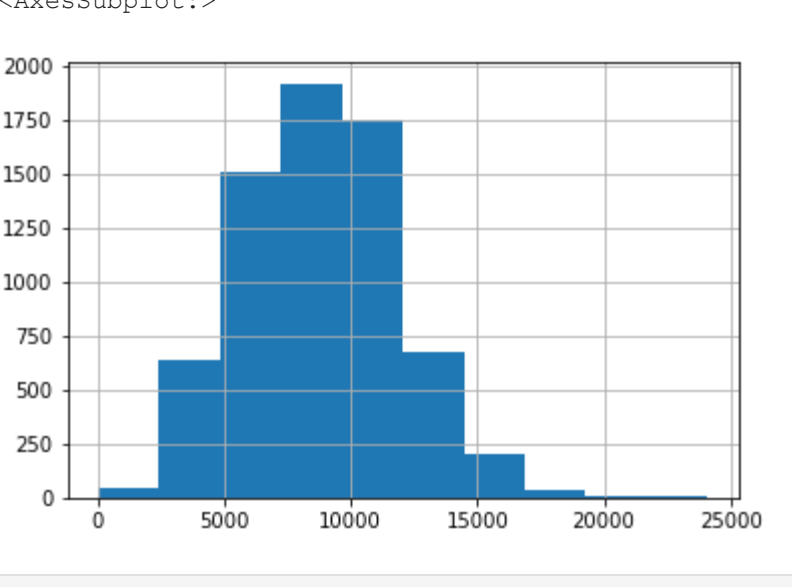


In [54]:
```python
scipy.stats.shapiro(data.loc[data['MMRCurrentAuctionCleanPrice'].isna()==False,'MMRCurrentAuctionCleanPrice'])
```

Out[54]:
```
ShapiroResult(statistic=0.9819827079772949, pvalue=1.7330095418579664e-28)
```

In [55]:
```python
data['MMRCurrentRetailAveragePrice'].hist()
```

Out[55]:
```
<AxesSubplot:>
```

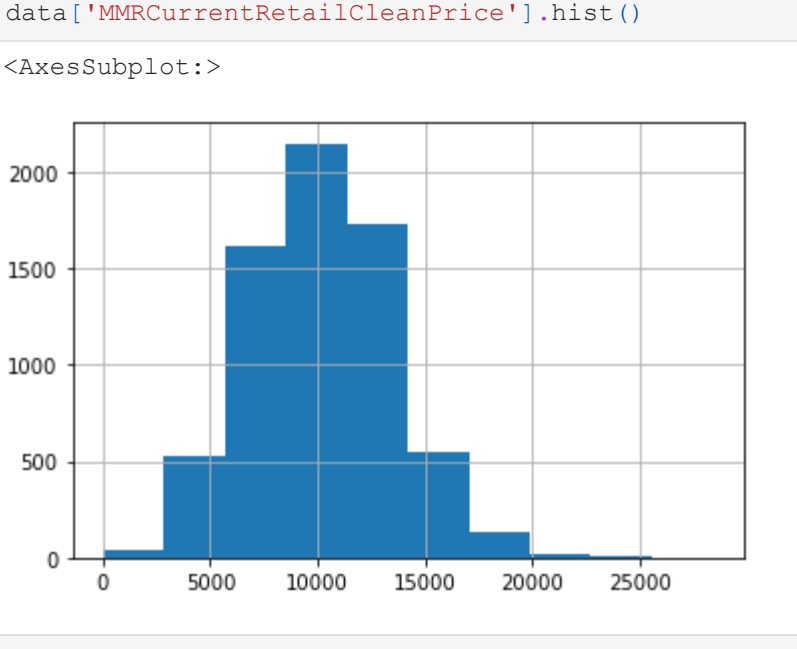


In [56]:
```python
scipy.stats.shapiro(data.loc[data['MMRCurrentRetailAveragePrice'].isna()==False,'MMRCurrentRetailAveragePrice']
```

Out[56]:
```
ShapiroResult(statistic=0.9944936633110046, pvalue=1.776779932357404e-15)
```

In [57]:
```python
data['MMRCurrentRetailCleanPrice'].hist()
```

Out[57]:
```
<AxesSubplot:>
```



In [58]:
```python
scipy.stats.shapiro(data.loc[data['MMRCurrentRetailCleanPrice'].isna()==False,'MMRCurrentRetailCleanPrice'])
```

Out[58]:
```
ShapiroResult(statistic=0.9937793612480164, pvalue=1.276671616205527e-16)
```

In [59]:
```python
# na osnovu histograma, pretpostavile smo da nijedna varijabla nema N raspodelu, ali kako bismo potvrdile
# uradile smo i Shapiro Wilk test i utvrdile da nemaju, tako da cemo nedostajuce vrednosti popuniti medijanom
```

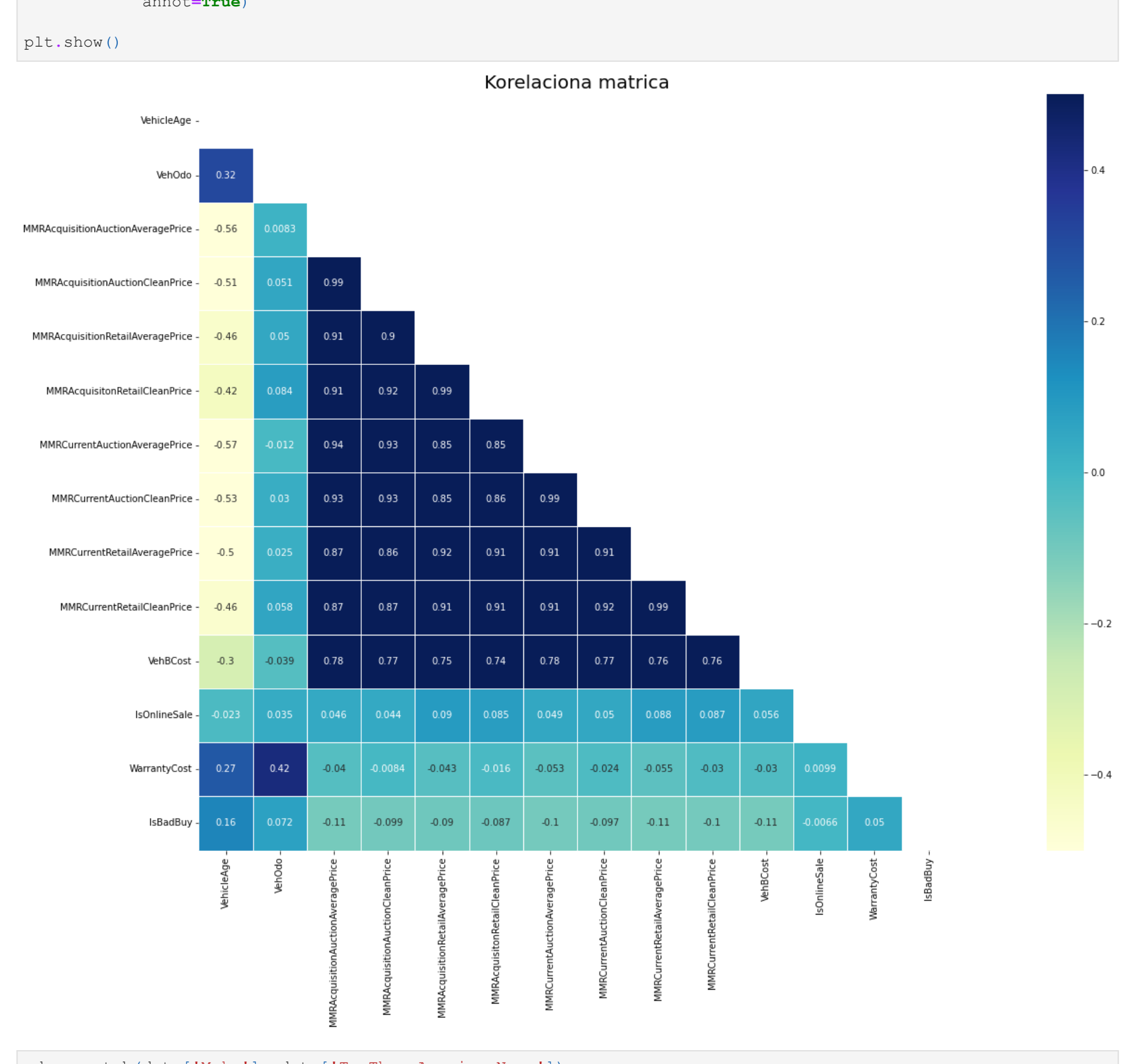
In [60]:
```python
data['MMRCurrentAuctionAveragePrice'] = data['MMRCurrentAuctionAveragePrice'].fillna(data['MMRCurrentAuctionAve
data['MMRCurrentAuctionCleanPrice'] = data['MMRCurrentAuctionCleanPrice'].fillna(data['MMRCurrentAuctionCleanPr
data['MMRCurrentRetailAveragePrice'] = data['MMRCurrentRetailAveragePrice'].fillna(data['MMRCurrentRetailAverag
data['MMRCurrentRetailCleanPrice'] = data['MMRCurrentRetailCleanPrice'].fillna(data['MMRCurrentRetailCleanPrice
```

In [61]:
```python
data.isna().sum()
```

Out[61]:
```
PurchDate                            0
VehicleAge                           0
Make                                 0
Model                                0
SubModel                             0
Color                                0
Transmission                         0
WheelType                            0
VehOdo                               0
Size                                 0
TopThreeAmericanName                 0
MMRAcquisitionAuctionAveragePrice    0
MMRAcquisitionAuctionCleanPrice      0
MMRAcquisitionRetailAveragePrice     0
MMRAcquisitonRetailCleanPrice        0
MMRCurrentAuctionAveragePrice        0
MMRCurrentAuctionCleanPrice          0
MMRCurrentRetailAveragePrice         0
MMRCurrentRetailCleanPrice           0
VehBCost                             0
IsOnlineSale                         0
WarrantyCost                         0
IsBadBuy                             0
dtype: int64
```

In [62]:
```python
# Sada je nas dataset sredjen i nemamo nedostajuce vrednosti, pa mozemo nastaviti sa daljom analizom
```

In [63]:
```python
corr = data.select_dtypes(np.number).corr()
plt.figure(figsize=(35,15))
plt.title('Korelaciona matrica', fontsize=20)

sns.heatmap(corr,
           vmin=-.5, #sve preko 0.5 (ispod -0.5) ce biti predstavljeno kao srednja i visoka korelisanost
           vmax =.5,
           mask = np.triu(np.ones_like(corr, dtype=np.bool)),
           cmap = "YlGnBu",
           center=0,
           square=True,
           linewidths=.5,
           annot=True)

plt.show()
```



In [64]:
```python
pd.crosstab(data['Make'], data['TopThreeAmericanName'])
```

Out[64]:

TopThreeAmericanName	CHRYSLER	FORD	GM	OTHER
Make				
ACURA	0	0	0	4
BUICK	0	0	63	0
CADILLAC	0	0	1	0
CHEVROLET	0	0	1574	0
CHRYSLER	827	0	0	0
DODGE	1187	0	0	0
FORD	0	1059	0	0
GMC	0	0	61	0
HONDA	0	0	0	52
HUMMER	0	0	1	0
HYUNDAI	0	0	0	181
INFINITI	0	0	0	6
ISUZU	0	0	0	8
JEEP	149	0	1	0
KIA	0	0	0	232
LEXUS	0	0	0	1
LINCOLN	0	7	0	0
MAZDA	0	0	0	112
MERCURY	0	81	0	0
MINI	0	0	0	5
MITSUBISHI	0	0	0	97
NISSAN	0	0	0	194
OLDSMOBILE	0	0	22	0
PLYMOUTH	1	0	0	0
PONTIAC	0	0	395	0
SATURN	0	0	211	0
SCION	0	0	0	5
SUBARU	0	0	0	1
SUZUKI	0	0	0	132
TOYOTA	0	0	0	109
VOLKSWAGEN	0	0	0	14
VOLVO	0	0	0	4

In [65]:
```python
# TopThreeAmericanName je takodje u visokoj korelaciji sa Make, tako da cemo i tu varijablu iskljuciti iz dalje
```

In [66]:
```python
data = data.drop('TopThreeAmericanName',axis=1)
```

In [67]:
```python
data.head()
```

Out[67]:

	PurchDate	VehicleAge	Make	Model	SubModel	Color	Transmission	WheelType	VehOdo	Size	MMRAcquisitionAuctio
RefId											
1	1/7/2009 0:12	4	DODGE	STRATUS V6	4D SEDAN SXT FFV	MAROON	AUTO	Covers	73807	MEDIUM	
2	1/7/2009 0:12	5	DODGE	NEON	4D SEDAN	SILVER	AUTO	Alloy	65617	COMPACT	
3	1/7/2009 0:12	4	FORD	FOCUS	2D COUPE ZX3	SILVER	MANUAL	Covers	69367	COMPACT	
4	1/7/2009 0:12	5	MITSUBISHI	GALANT 4C	4D SEDAN ES	WHITE	AUTO	Covers	81054	MEDIUM	
5	1/21/2009 0:12	5	DODGE	1500 RAM PICKUP 2WD	QUAD CAB 4.7L SLT	WHITE	AUTO	Alloy	76173	LARGE TRUCK	

In [68]:
```
# Analiza ostalih kategorickih atributa
```

In [69]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6797 entries, 1 to 6798
Data columns (total 22 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   PurchDate                          6797 non-null   object
 1   VehicleAge                         6797 non-null   int64
 2   Make                               6797 non-null   object
 3   Model                              6797 non-null   object
 4   SubModel                           6797 non-null   object
 5   Color                              6797 non-null   object
 6   Transmission                       6797 non-null   object
 7   WheelType                          6797 non-null   object
 8   VehOdo                             6797 non-null   int64
 9   Size                               6797 non-null   object
 10  MMRAcquisitionAuctionAveragePrice  6797 non-null   float64
 11  MMRAcquisitionAuctionCleanPrice    6797 non-null   float64
 12  MMRAcquisitionRetailAveragePrice   6797 non-null   float64
 13  MMRAcquisitonRetailCleanPrice      6797 non-null   float64
 14  MMRCurrentAuctionAveragePrice      6797 non-null   float64
 15  MMRCurrentAuctionCleanPrice        6797 non-null   float64
 16  MMRCurrentRetailAveragePrice       6797 non-null   float64
 17  MMRCurrentRetailCleanPrice         6797 non-null   float64
 18  VehBCost                           6797 non-null   float64
 19  IsOnlineSale                       6797 non-null   int64
 20  WarrantyCost                       6797 non-null   int64
 21  IsBadBuy                           6797 non-null   int64
dtypes: float64(9), int64(5), object(8)
memory usage: 1.2+ MB
```

In [70]:
```
# Purchdate
```

In [71]:
```
data['PurchDate'].value_counts()
```

Out[71]:
```
1/30/2010 0:09    42
1/25/2009 0:02    41
1/20/2010 0:10    40
1/4/2009 0:03     36
1/8/2010 0:12     36
                  ..
1/16/2009 0:01     1
1/26/2010 0:03     1
1/19/2010 0:04     1
1/27/2010 0:09     1
1/25/2009 0:05     1
Name: PurchDate, Length: 503, dtype: int64
```

In [72]:
```
# Za nasu analizu znacajna nam je godina aukcije, tako da cemo napraviti novu varijablu gde cemo iz ove izvuci
```

In [73]:
```
data['PurchYear'] = pd.DatetimeIndex(data['PurchDate']).year
```

In [74]:
```
data = data.drop('PurchDate',axis=1)
```

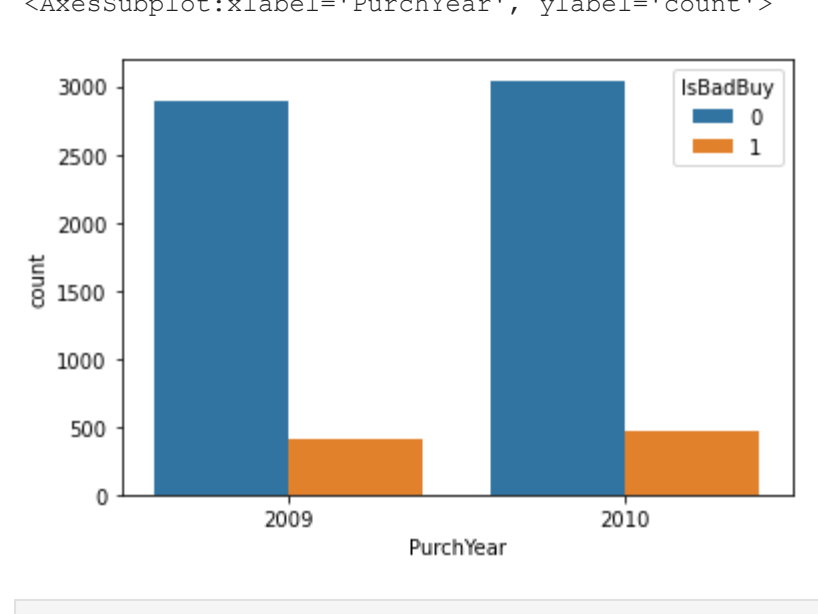
In [75]:
```
data['PurchYear'].value_counts()
```

Out[75]:
```
2010    3504
2009    3293
Name: PurchYear, dtype: int64
```

In [76]:
```
# Vidimo da imamo samo dve godine i da je broj opservacija priblizno isti za obe, pa cemo sada proveriti kakva
# u odnosu na izlaznu varijablu
```

In [77]:
```
sns.countplot(x='PurchYear', hue='IsBadBuy', data=data)
```

Out[77]:
```
<AxesSubplot:xlabel='PurchYear', ylabel='count'>
```



In [78]:
```
# Zakljucujemo da ne postoji znacajna razlika u kvalitetu kupovine u odnosu na ove dve godine, ali ce nam kombi
# ovog atributa sa nekim drugim mozda dati bolje rezultate, tako da cemo je za sada ostaviti u datasetu
```

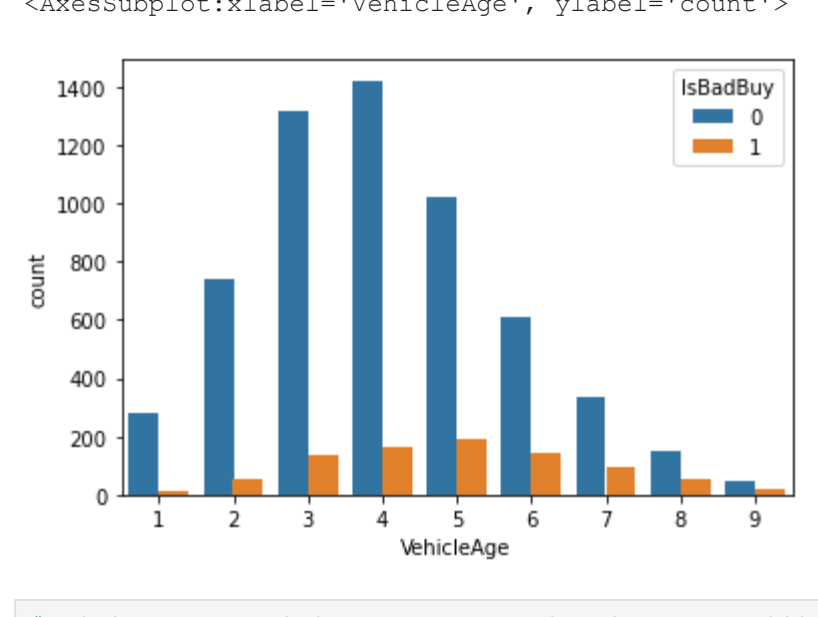
In [79]:
```
# VehicleAge
```

In [80]:
```
data['VehicleAge'].value_counts()
```

Out[80]:
```
4    1588
3    1453
5    1214
2     792
6     753
7     429
1     295
8     207
9      66
Name: VehicleAge, dtype: int64
```

In [81]:
```
sns.countplot(x='VehicleAge', hue='IsBadBuy', data=data)
```

Out[81]:
```
<AxesSubplot:xlabel='VehicleAge', ylabel='count'>
```



In [82]:
```
# Vidimo da razlicita starost vozila ima drugaciji uticaj na kupovinu, pa mozemo da zakljucimo da nam je ovaj a
# znacajan za dalju analizu
```

In [83]:
```
# Make
```

In [84]:
```
data['Make'].value_counts()
```

Out[84]:
```
CHEVROLET     1574
DODGE         1187
FORD          1059
CHRYSLER       827
PONTIAC        395
KIA            232
SATURN         211
NISSAN         194
HYUNDAI        181
JEEP           150
SUZUKI         132
MAZDA          112
TOYOTA         109
MITSUBISHI      97
MERCURY         81
BUICK           63
GMC             61
HONDA           52
OLDSMOBILE      22
VOLKSWAGEN      14
ISUZU            8
LINCOLN          7
INFINITI         6
MINI             5
SCION            5
VOLVO            4
ACURA            4
SUBARU           1
CADILLAC         1
HUMMER           1
PLYMOUTH         1
LEXUS            1
Name: Make, dtype: int64
```

In [85]:
```
# Model
```

In [86]:
```
data['Model'].value_counts()
```

Out[86]:
```
PT CRUISER              216
IMPALA                  194
TAURUS                  128
CARAVAN GRAND FWD V6    118
CALIBER                 117
                       ...
CAMRY V6 3.0L / 3.3L      1
MARINER 2WD 4C            1
SILHOUETTE                1
VERONA                    1
PATRIOT 2WD 4C 2.0L       1
Name: Model, Length: 632, dtype: int64
```

In [87]:
```
pd.crosstab(data['Model'], data['SubModel']).sum()
```

Out[87]:
```
SubModel
2D CONVERTIBLE             30
2D CONVERTIBLE GLS          1
2D CONVERTIBLE GTC          2
2D CONVERTIBLE TOURING      7
2D COUPE                  102
                         ...
WAGON 3.5L                  2
WAGON 3.5L SXT              3
WAGON DX                    1
WAGON LX                    1
WAGON R/T                   1
Length: 465, dtype: int64
```

In [88]:
```
# Vidimo da za oba varijable imamo veliki broj razlicitih vrednosti, pa cemo obe varijable iskljuciti iz dalje
```

In [89]:
```
data.drop(['Model', 'SubModel'], axis=1, inplace=True)
```

In [90]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6797 entries, 1 to 6798
Data columns (total 20 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   VehicleAge                         6797 non-null   int64
 1   Make                               6797 non-null   object
 2   Color                              6797 non-null   object
 3   Transmission                       6797 non-null   object
 4   WheelType                          6797 non-null   object
 5   VehOdo                             6797 non-null   int64
 6   Size                               6797 non-null   object
 7   MMRAcquisitionAuctionAveragePrice  6797 non-null   float64
 8   MMRAcquisitionAuctionCleanPrice    6797 non-null   float64
 9   MMRAcquisitionRetailAveragePrice   6797 non-null   float64
 10  MMRAcquisitonRetailCleanPrice      6797 non-null   float64
 11  MMRCurrentAuctionAveragePrice      6797 non-null   float64
 12  MMRCurrentAuctionCleanPrice        6797 non-null   float64
 13  MMRCurrentRetailAveragePrice       6797 non-null   float64
 14  MMRCurrentRetailCleanPrice         6797 non-null   float64
 15  VehBCost                           6797 non-null   float64
 16  IsOnlineSale                       6797 non-null   int64
 17  WarrantyCost                       6797 non-null   int64
 18  IsBadBuy                           6797 non-null   int64
 19  PurchYear                          6797 non-null   int64
dtypes: float64(9), int64(6), object(5)
memory usage: 1.1+ MB
```

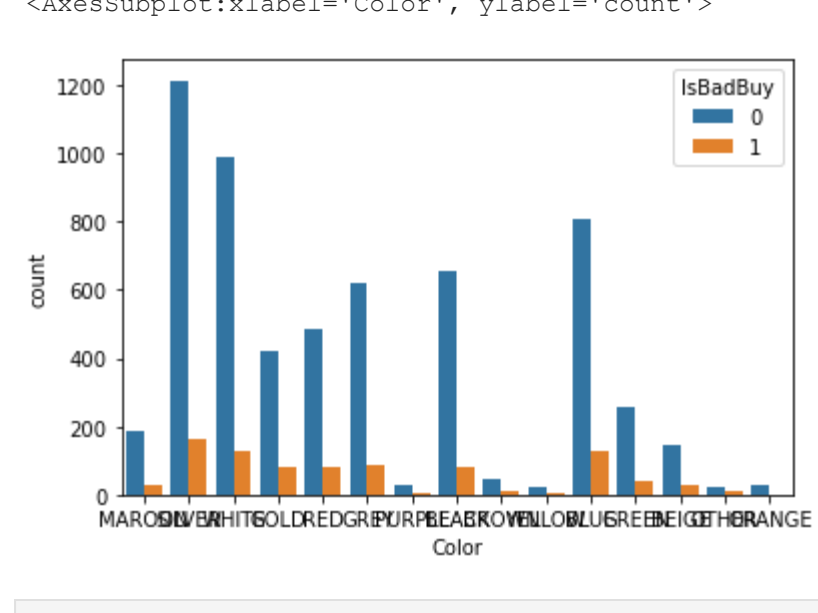
In [91]:
```
# Color
```

In [92]:
```
data['Color'].value_counts()
```

Out[92]:
```
SILVER       1379
WHITE        1118
BLUE          930
BLACK         735
GREY          710
RED           570
GOLD          504
GREEN         297
MAROON        210
BEIGE         170
BROWN          52
PURPLE         32
ORANGE         29
YELLOW         28
OTHER          28
NOT AVAIL       5
Name: Color, dtype: int64
```

In [314…
```
sns.countplot(x='Color', hue='IsBadBuy', data=data)
```

Out[314]:
```
<AxesSubplot:xlabel='Color', ylabel='count'>
```



In [94]:
```
data.loc[data['Color'] == 'NOT AVAIL','Color']
```

Out[94]:
```
RefId
2918    NOT AVAIL
4072    NOT AVAIL
4160    NOT AVAIL
4910    NOT AVAIL
4912    NOT AVAIL
Name: Color, dtype: object
```

In [95]:
```
# Vidimo da za 5 opservacija imamo vrednost not avail, sto moramo zameniti
# Medjutim, necemo menjati najzastupljenijom bojom, vec cemo im dodeliti vrednost other
```

In [96]:
```
data.loc[data['Color'] == 'NOT AVAIL','Color'] = 'OTHER'
```

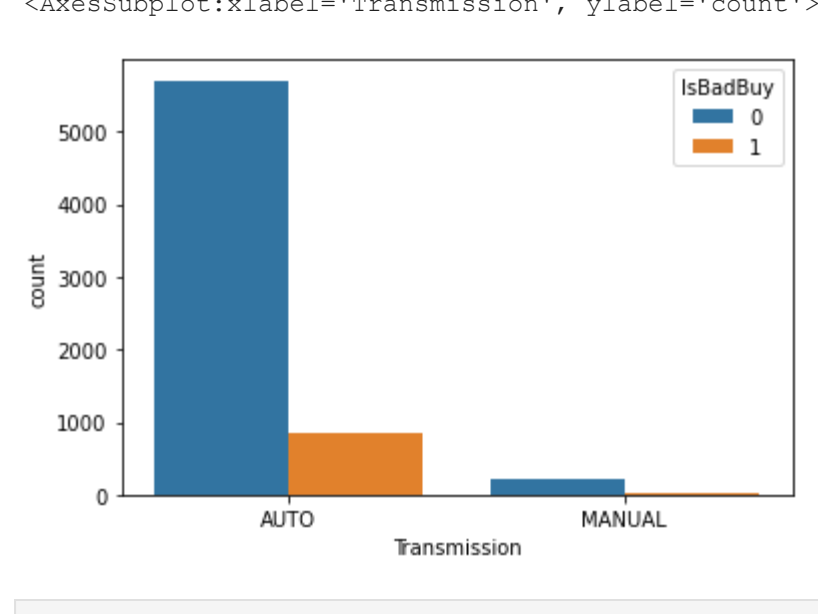
In [97]:
```
# Transmission
```

In [98]:
```
data['Transmission'].value_counts()
```

Out[98]:
```
AUTO      6546
MANUAL     251
Name: Transmission, dtype: int64
```

In [99]:
```
sns.countplot(x='Transmission', hue='IsBadBuy', data=data)
```

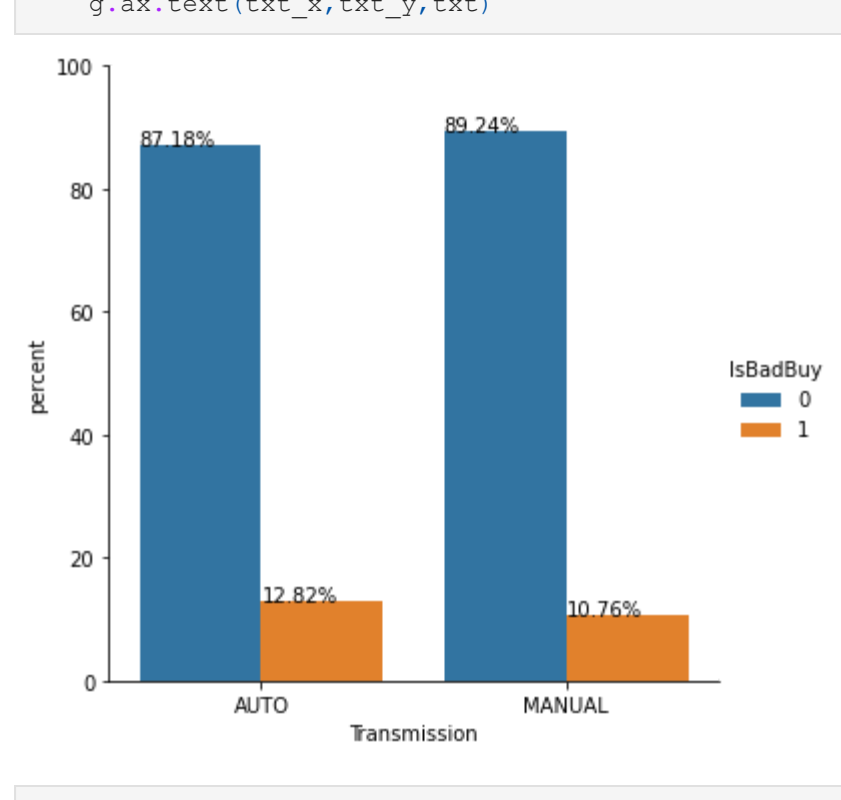
Out[99]:
```
<AxesSubplot:xlabel='Transmission', ylabel='count'>
```



In [100…
```
#Zbog velike razlike u broju opservacija koje pripadaju ovim kategorijama, na osnovu ovog dijagrama ne mozemo j
#utvrdimo da li ima znacajne razlike izmedju automatika i manuala, pa cemo izraziti procentualno
```

In [101…
```
dataTrans = data.groupby('Transmission')['IsBadBuy'].value_counts(normalize=True)
dataTrans = dataTrans.mul(100)
dataTrans = dataTrans.rename('percent').reset_index()
```

In [102…
```
g = sns.catplot(x='Transmission', y='percent', hue='IsBadBuy', kind='bar', data=dataTrans)
g.ax.set_ylim(0,100)
for p in g.ax.patches:
    txt = str(p.get_height().round(2)) + '%'
    txt_x = p.get_x()
    txt_y = p.get_height()
    g.ax.text(txt_x,txt_y,txt)
```



In [103…
```
# Uocavamo da ne postoji znacajna razlika izmedju kvaliteta kupovine u zavisnosti od kategorije kojoj vozilo pr
#ali ce mozda ovaj atribut biti znacajan u kombinaciji sa nekim drugim pa ga zato za sada ostavljamo u analizi
```

In [104…
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6797 entries, 1 to 6798
Data columns (total 20 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   VehicleAge                         6797 non-null   int64
 1   Make                               6797 non-null   object
 2   Color                              6797 non-null   object
 3   Transmission                       6797 non-null   object
 4   WheelType                          6797 non-null   object
 5   VehOdo                             6797 non-null   int64
 6   Size                               6797 non-null   object
 7   MMRAcquisitionAuctionAveragePrice  6797 non-null   float64
 8   MMRAcquisitionAuctionCleanPrice    6797 non-null   float64
 9   MMRAcquisitionRetailAveragePrice   6797 non-null   float64
 10  MMRAcquisitonRetailCleanPrice      6797 non-null   float64
 11  MMRCurrentAuctionAveragePrice      6797 non-null   float64
 12  MMRCurrentAuctionCleanPrice        6797 non-null   float64
 13  MMRCurrentRetailAveragePrice       6797 non-null   float64
 14  MMRCurrentRetailCleanPrice         6797 non-null   float64
 15  VehBCost                           6797 non-null   float64
 16  IsOnlineSale                       6797 non-null   int64
 17  WarrantyCost                       6797 non-null   int64
 18  IsBadBuy                           6797 non-null   int64
 19  PurchYear                          6797 non-null   int64
dtypes: float64(9), int64(6), object(5)
memory usage: 1.1+ MB
```

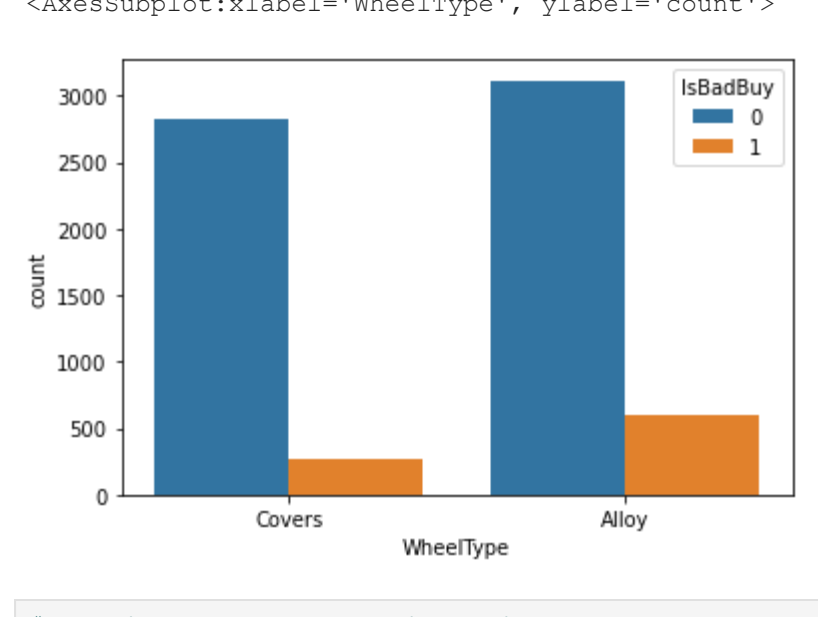
In [105…
```
#WheelType
```

In [106…
```
data['WheelType'].value_counts()
```

Out[106]:
```
Alloy     3714
Covers    3083
Name: WheelType, dtype: int64
```

In [107…
```
sns.countplot(x='WheelType', hue='IsBadBuy', data=data)
```

Out[107]:
```
<AxesSubplot:xlabel='WheelType', ylabel='count'>
```



In [108…
```
# Kao i prethodna, ostavicemo je
```

In [109…
```
data['IsBadBuy'].value_counts()
```

Out[109]:
```
0    5931
1     866
Name: IsBadBuy, dtype: int64
```

In [110…
```
# Vidimo da izlazna varijabla ima nebalansirane klase, sto je lose, resicemo
```

In [111…
```
#Size
```

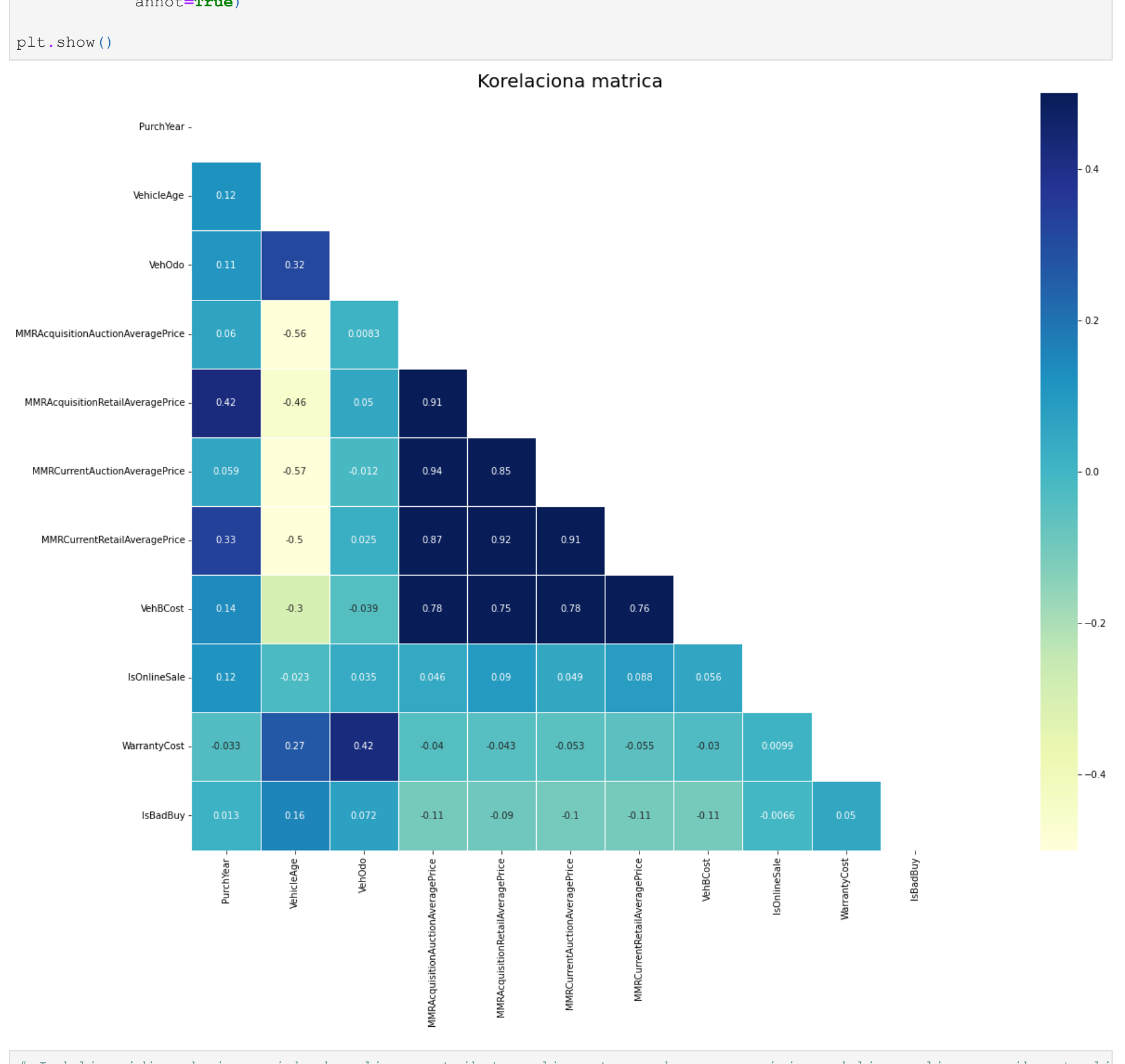
In [112…
```
data['Size'].value_counts()
```

Out[112]:
```
MEDIUM         2889
LARGE           786
MEDIUM SUV      760
COMPACT         629
VAN             550
LARGE TRUCK     322
SMALL SUV       221
CROSSOVER       174
SPECIALTY       169
LARGE SUV       134
SMALL TRUCK      90
SPORTS           73
Name: Size, dtype: int64
```

In [113…
```
kolone = ['PurchYear', 'VehicleAge', 'Make', 'Color', 'Transmission', 'WheelType',
          'VehOdo', 'Size', 'MMRAcquisitionAuctionAveragePrice',
          'MMRAcquisitionRetailAveragePrice', 'MMRCurrentAuctionAveragePrice', 'MMRCurrentRetailAverag
          'VehBCost', 'IsOnlineSale', 'WarrantyCost', 'IsBadBuy']
data_prosek = data[kolone]
```

In [114…
```
# Posto vidimo da su MMR vrednosti u visokoj korelaciji, napravicemo data set gde cemo izbaciti 4 varijable koj
# na cenu automobila koji je u natprosecnom stanju, pa cemo testirati na oba data seta (sa i bez) i videti sta
```

In [115…
```
corr = data_prosek.select_dtypes(np.number).corr()
plt.figure(figsize=(35,15))
plt.title('Korelaciona matrica', fontsize=20)

sns.heatmap(corr,
            vmin=-.5, #sve preko 0.5 (ispod -0.5) ce biti predstavljeno kao srednja i visoka korelisanost
            vmax =.5,
            mask = np.triu(np.ones_like(corr, dtype=np.bool)),
            cmap = "YlGnBu",
            center=0,
            square=True,
            linewidths=.5,
            annot=True)

plt.show()
```



In [116…
```
# I dalje vidimo da imamo jako korelisane atribute, ali smatramo da su znacajni za dalju analizu, pa ih ostavlj
```

In [117…
```
# IsOnlineSale
```

In [118…
```
data['IsOnlineSale'].value_counts()
```

Out[118]:
```
0    6613
1     184
Name: IsOnlineSale, dtype: int64
```

In [119…
```
# Vidimo da velika vecina pripada klasi da nije online, tako da cemo je izbaciti iz dalje analize
```

In [120…
```
data = data.drop("IsOnlineSale",axis=1)
```

In [121…
```
data_prosek = data_prosek.drop("IsOnlineSale",axis=1)
```

Kvantifikacija kategorickih atributa

In [123…
```
# Gde imamo samo dve kategorije, dodelicemo binarne vrednosti, a za varijable sa vise kategorija radimo dummy
```

In [124…
```
# Izabrale smo da radimo sa data.prosek pa cemo videti rezultate na kraju i odluciti da li je bio dobar izbor i
```

In [125…
```
data_prosek = data_prosek.replace({'Transmission' : {'MANUAL':0, 'AUTO':1}})
data_prosek = data_prosek.replace({'WheelType' : {'Covers':0, 'Alloy':1}})
```

In [126…
```
data_prosek = pd.get_dummies(data_prosek, columns = ['Make', 'Color', 'Size'])
data_prosek
```

Out[126]:

	PurchYear	VehicleAge	Transmission	WheelType	VehOdo	MMRAcquisitionAuctionAveragePrice	MMRAcquisitionRetailAveragePrice	MMI
RefId								
1	2009	4	1	0	73807	3202.0	6943.0	
2	2009	5	1	1	65617	1893.0	4658.0	
3	2009	4	0	0	69367	3913.0	7723.0	
4	2009	5	1	0	81054	3901.0	6706.0	
5	2009	5	1	1	76173	6407.0	10378.0	
...	...	...	...	...	...	...	...	
6794	2009	3	1	0	60549	3905.0	4717.0	
6795	2009	4	1	1	85377	7104.0	8172.0	
6796	2009	7	1	1	75700	5095.0	6003.0	
6797	2009	3	1	0	69941	12195.0	13671.0	
6798	2009	7	1	1	93744	5485.0	6424.0	

6797 rows × 71 columns

In [127…
```
# Delimo dataset na izlaznu i ostale promenljive
```

In [128…
```
data_prosek['IsBadBuy'] = data_prosek['IsBadBuy'].astype('object')
```

In [129…
```
x = data_prosek.loc[:, data_prosek.columns != 'IsBadBuy']
y = data_prosek.loc[:, 'IsBadBuy']
```

In [130…
```
x.head()
y.dtypes
```

Out[130]:
```
dtype('O')
```

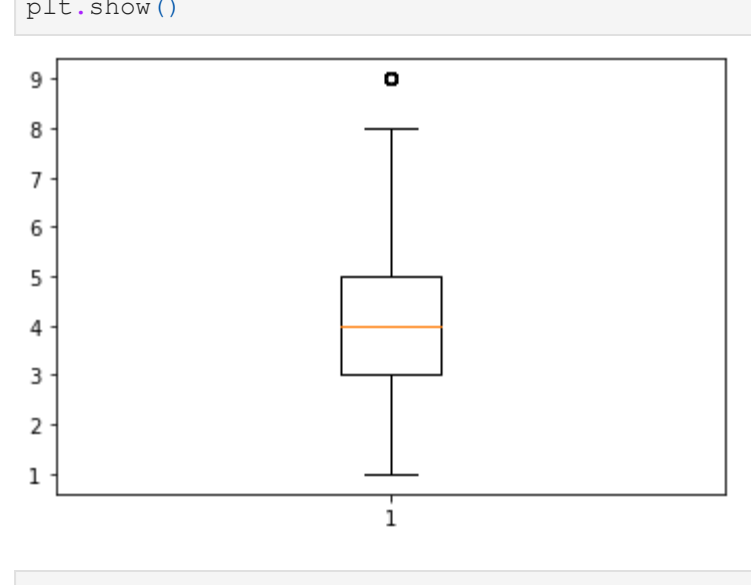
In [133…
```
# Vidimo da smo uspeli da smanjimo broj varijabli sa 70 na 27, sto znaci da je dosta njih bilo beznacajno za na
```

In [134…
```
# Sredjivanje autlajera
```

In [135…
```
data_prosek.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6797 entries, 1 to 6798
Data columns (total 71 columns):
 #   Column                             Non-Null Count  Dtype
---  ------                             --------------  -----
 0   PurchYear                          6797 non-null   int64
 1   VehicleAge                         6797 non-null   int64
 2   Transmission                       6797 non-null   int64
 3   WheelType                          6797 non-null   int64
 4   VehOdo                             6797 non-null   int64
 5   MMRAcquisitionAuctionAveragePrice  6797 non-null   float64
 6   MMRAcquisitionRetailAveragePrice   6797 non-null   float64
 7   MMRCurrentAuctionAveragePrice      6797 non-null   float64
 8   MMRCurrentRetailAveragePrice       6797 non-null   float64
 9   VehBCost                           6797 non-null   float64
 10  WarrantyCost                       6797 non-null   int64
 11  IsBadBuy                           6797 non-null   object
 12  Make_ACURA                         6797 non-null   uint8
 13  Make_BUICK                         6797 non-null   uint8
 14  Make_CADILLAC                      6797 non-null   uint8
 15  Make_CHEVROLET                     6797 non-null   uint8
 16  Make_CHRYSLER                      6797 non-null   uint8
 17  Make_DODGE                         6797 non-null   uint8
 18  Make_FORD                          6797 non-null   uint8
 19  Make_GMC                           6797 non-null   uint8
 20  Make_HONDA                         6797 non-null   uint8
 21  Make_HUMMER                        6797 non-null   uint8
 22  Make_HYUNDAI                       6797 non-null   uint8
 23  Make_INFINITI                      6797 non-null   uint8
 24  Make_ISUZU                         6797 non-null   uint8
 25  Make_JEEP                          6797 non-null   uint8
 26  Make_KIA                           6797 non-null   uint8
 27  Make_LEXUS                         6797 non-null   uint8
 28  Make_LINCOLN                       6797 non-null   uint8
 29  Make_MAZDA                         6797 non-null   uint8
 30  Make_MERCURY                       6797 non-null   uint8
 31  Make_MINI                          6797 non-null   uint8
 32  Make_MITSUBISHI                    6797 non-null   uint8
 33  Make_NISSAN                        6797 non-null   uint8
 34  Make_OLDSMOBILE                    6797 non-null   uint8
 35  Make_PLYMOUTH                      6797 non-null   uint8
 36  Make_PONTIAC                       6797 non-null   uint8
 37  Make_SATURN                        6797 non-null   uint8
 38  Make_SCION                         6797 non-null   uint8
 39  Make_SUBARU                        6797 non-null   uint8
 40  Make_SUZUKI                        6797 non-null   uint8
 41  Make_TOYOTA                        6797 non-null   uint8
 42  Make_VOLKSWAGEN                    6797 non-null   uint8
 43  Make_VOLVO                         6797 non-null   uint8
 44  Color_BEIGE                        6797 non-null   uint8
 45  Color_BLACK                        6797 non-null   uint8
 46  Color_BLUE                         6797 non-null   uint8
 47  Color_BROWN                        6797 non-null   uint8
 48  Color_GOLD                         6797 non-null   uint8
 49  Color_GREEN                        6797 non-null   uint8
 50  Color_GREY                         6797 non-null   uint8
 51  Color_MAROON                       6797 non-null   uint8
 52  Color_ORANGE                       6797 non-null   uint8
 53  Color_OTHER                        6797 non-null   uint8
 54  Color_PURPLE                       6797 non-null   uint8
 55  Color_RED                          6797 non-null   uint8
 56  Color_SILVER                       6797 non-null   uint8
 57  Color_WHITE                        6797 non-null   uint8
 58  Color_YELLOW                       6797 non-null   uint8
 59  Size_COMPACT                       6797 non-null   uint8
 60  Size_CROSSOVER                     6797 non-null   uint8
 61  Size_LARGE                         6797 non-null   uint8
 62  Size_LARGE SUV                     6797 non-null   uint8
 63  Size_LARGE TRUCK                   6797 non-null   uint8
 64  Size_MEDIUM                        6797 non-null   uint8
 65  Size_MEDIUM SUV                    6797 non-null   uint8
 66  Size_SMALL SUV                     6797 non-null   uint8
 67  Size_SMALL TRUCK                   6797 non-null   uint8
 68  Size_SPECIALTY                     6797 non-null   uint8
 69  Size_SPORTS                        6797 non-null   uint8
 70  Size_VAN                           6797 non-null   uint8
dtypes: float64(5), int64(6), object(1), uint8(59)
memory usage: 1.1+ MB
```

In [136…
```
plt.boxplot(data_prosek['VehicleAge'])
plt.show()
```

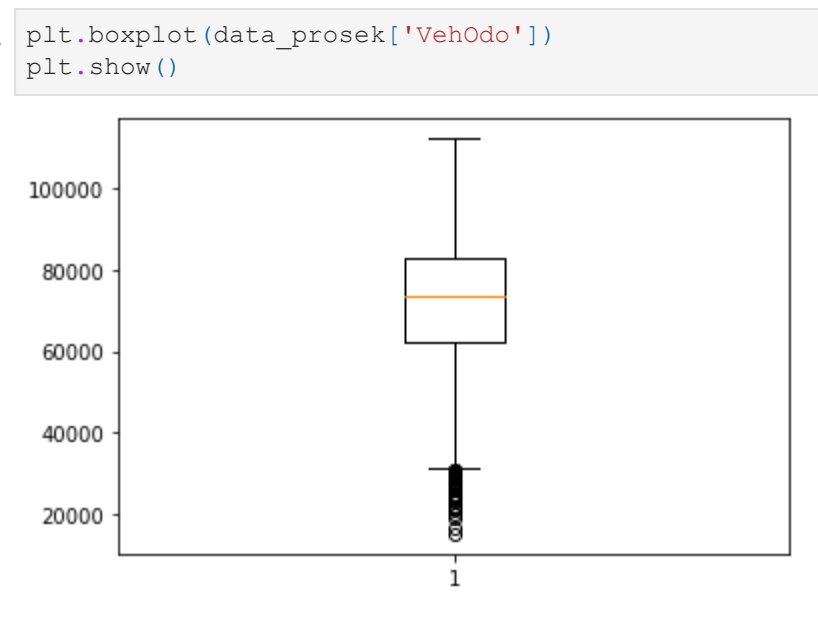


In [137…
```
Q1 = data_prosek['VehicleAge'].quantile(0.25)
Q3 = data_prosek['VehicleAge'].quantile(0.75)

# calculate the IQR
IQR = Q3 - Q1

# filter the dataset with the IQR
IQR_outliers = data_prosek['VehicleAge'][((data_prosek['VehicleAge'] < (Q1 - 1.5 * IQR)) |(data_prosek['Vehicle
IQR_outliers
```

Out[137]:
```
RefId
129     9
134     9
154     9
577     9
784     9
       ..
6436    9
6545    9
6602    9
6697    9
6717    9
Name: VehicleAge, Length: 66, dtype: int64
```

In [138…
```
data_prosek.loc[data_prosek['VehicleAge'] > (Q3 + 1.5 * IQR), 'VehicleAge'] = (Q3 + 1.5 * IQR)
```

In [139…
```
plt.boxplot(data_prosek['VehOdo'])
plt.show()
```

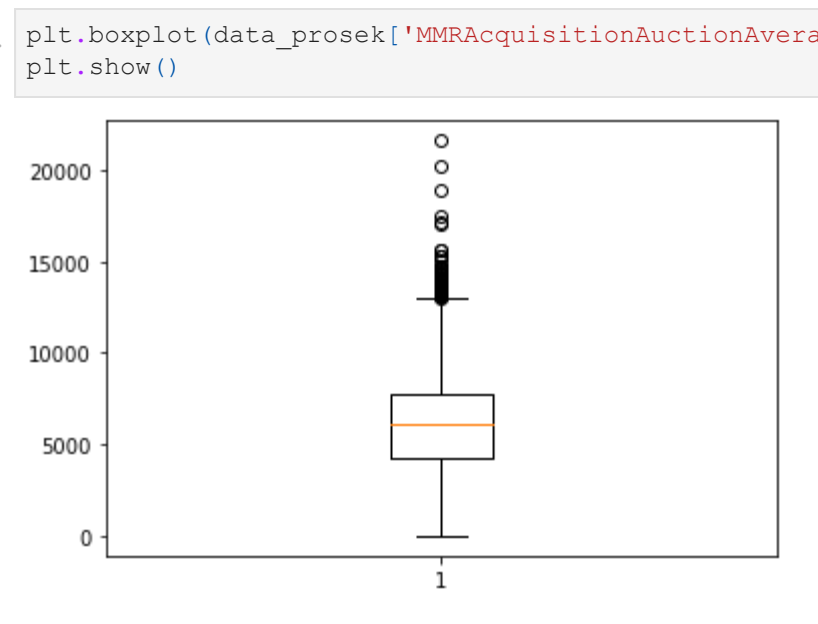


In [140…
```
Q1 = data_prosek['VehOdo'].quantile(0.25)
Q3 = data_prosek['VehOdo'].quantile(0.75)

# calculate the IQR
IQR = Q3 - Q1

# filter the dataset with the IQR
IQR_outliers = data_prosek['VehOdo'][((data_prosek['VehOdo'] < (Q1 - 1.5 * IQR)) |(data_prosek['VehOdo'] > (Q3
IQR_outliers
```

Out[140]:
```
RefId
42      25830
454     27504
459     19983
598     24186
619     30259
698     30128
825     21814
936     28802
1471    27815
1777    26524
1779    31118
1782    16574
2243    24805
2540    23144
2630    29314
3010    15249
3047    30584
3223    29084
3265    30807
3364    21459
3366    30553
3383    27281
3820    23376
4475    27115
5027    19070
5028    26811
5123    30868
5260    24481
5293    29431
5294    29335
5304    22241
5474    21158
5917    28184
6208    24735
6300    26625
6385    26660
6426    27247
6599    27308
Name: VehOdo, dtype: int64
```

In [141…
```
data_prosek.loc[data_prosek['VehOdo'] < (Q1 - 1.5 * IQR), 'VehOdo'] = (Q1 - 1.5 * IQR)
```

In [142…
```
plt.boxplot(data_prosek['MMRAcquisitionAuctionAveragePrice'])
plt.show()
```

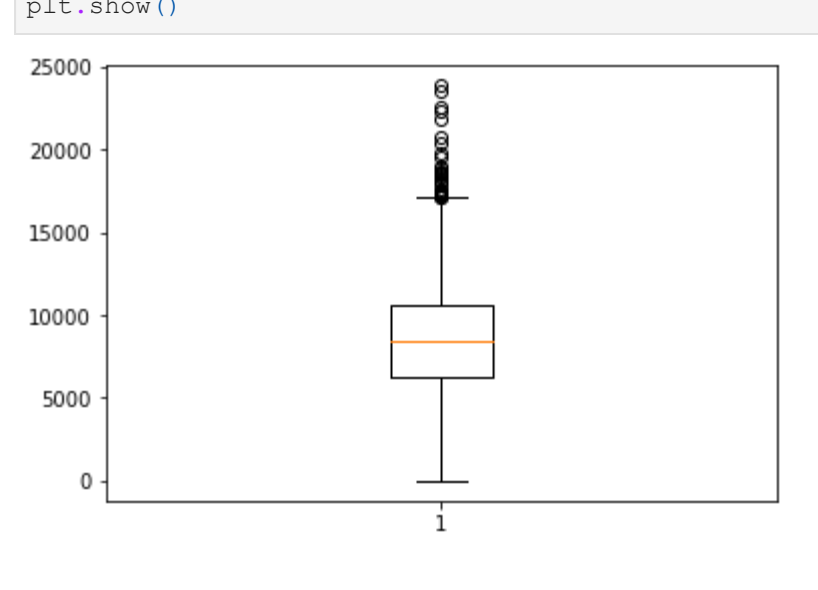


In [143…
```
Q1 = data_prosek['MMRAcquisitionAuctionAveragePrice'].quantile(0.25)
Q3 = data_prosek['MMRAcquisitionAuctionAveragePrice'].quantile(0.75)

# calculate the IQR
IQR = Q3 - Q1

# filter the dataset with the IQR
IQR_outliers = data_prosek['MMRAcquisitionAuctionAveragePrice'][((data_prosek['MMRAcquisitionAuctionAveragePric
IQR_outliers
```

Out[143]:
```
RefId
43      17110.0
216     13194.0
398     13572.0
729     13030.0
842     13106.0
         ...
6048    13940.0
6224    13670.0
6294    15599.0
6353    14467.0
6607    13150.0
Name: MMRAcquisitionAuctionAveragePrice, Length: 53, dtype: float64
```

In [144…
```
data_prosek.loc[data_prosek['MMRAcquisitionAuctionAveragePrice'] > (Q3 + 1.5 * IQR), 'MMRAcquisitionAuctionAver
```

In [145…
```
plt.boxplot(data_prosek['MMRAcquisitionRetailAveragePrice'])
plt.show()
```

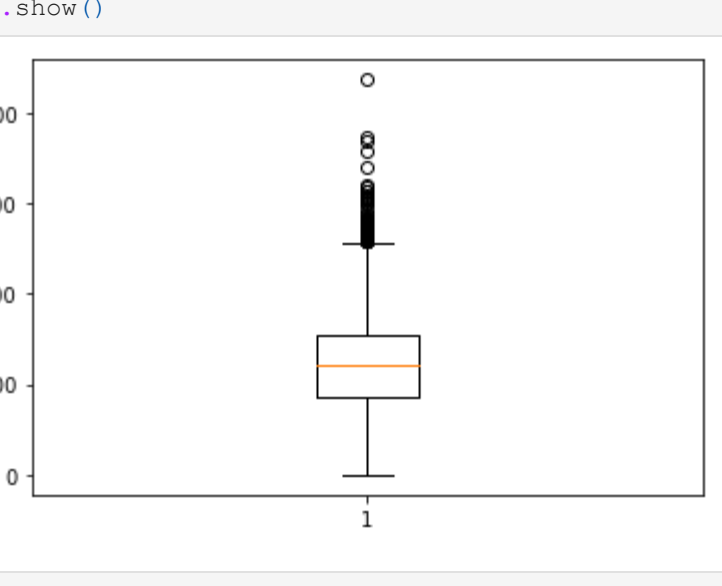


```
In [146]: Q1 = data_prosek['MMRAcquisitionRetailAveragePrice'].quantile(0.25)
Q3 = data_prosek['MMRAcquisitionRetailAveragePrice'].quantile(0.75)

# calculate the IQR
IQR = Q3 - Q1

# filter the dataset with the IQR
IQR_outliers = data_prosek['MMRAcquisitionRetailAveragePrice'][((data_prosek['MMRAcquisitionRetailAveragePrice'
IQR_outliers
```

```
Out[146]:
RefId
9       17400.0
43      21858.0
398     17365.0
1047    17321.0
1053    17131.0
1088    19736.0
1108    22536.0
1192    18495.0
1547    18555.0
1784    18910.0
2160    18658.0
2485    17121.0
2524    17408.0
2626    18275.0
3070    23840.0
3408    18907.0
3419    22357.0
3487    18073.0
3870    18515.0
4386    20737.0
4836    17195.0
4892    17301.0
4894    23456.0
5173    17327.0
5532    17608.0
5533    18131.0
5617    20371.0
6046    17526.0
6048    19560.0
6071    17266.0
6224    17986.0
6294    17347.0
6353    18611.0
6607    17516.0
Name: MMRAcquisitionRetailAveragePrice, dtype: float64
```

```
In [147]: data_prosek.loc[data_prosek['MMRAcquisitionRetailAveragePrice'] > (Q3 + 1.5 * IQR), 'MMRAcquisitionRetailAverag
```

```
In [148]: plt.boxplot(data_prosek['MMRCurrentAuctionAveragePrice'])
plt.show()
```

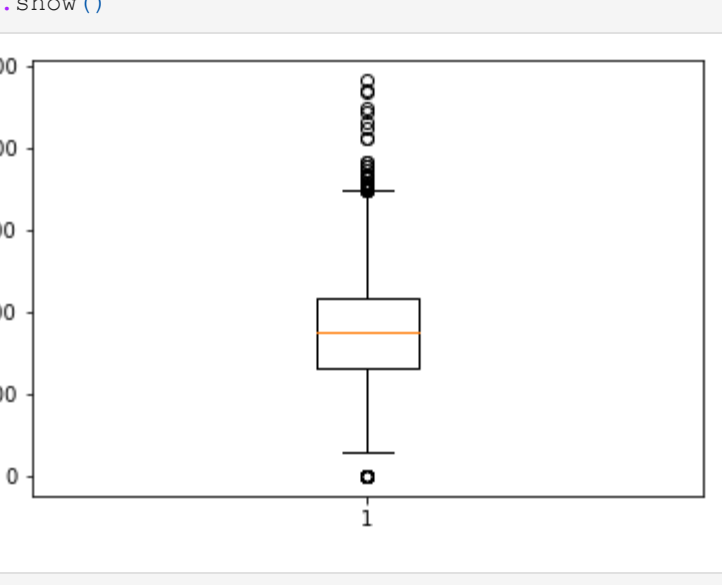


```
In [149]: Q1 = data_prosek['MMRCurrentAuctionAveragePrice'].quantile(0.25)
Q3 = data_prosek['MMRCurrentAuctionAveragePrice'].quantile(0.75)

# calculate the IQR
IQR = Q3 - Q1

# filter the dataset with the IQR
IQR_outliers = data_prosek['MMRCurrentAuctionAveragePrice'][((data_prosek['MMRCurrentAuctionAveragePrice'] < (Q
IQR_outliers
```

```
Out[149]:
RefId
43      16043.0
216     13278.0
342     12942.0
398     13256.0
655     12958.0
         ...
6296    13237.0
6464    13938.0
6472    13717.0
6607    13277.0
6797    13646.0
Name: MMRCurrentAuctionAveragePrice, Length: 71, dtype: float64
```

```
In [150]: data_prosek.loc[data_prosek['MMRCurrentAuctionAveragePrice'] > (Q3 + 1.5 * IQR), 'MMRCurrentAuctionAveragePrice
```

```
In [151]: plt.boxplot(data_prosek['MMRCurrentRetailAveragePrice'])
plt.show()
```



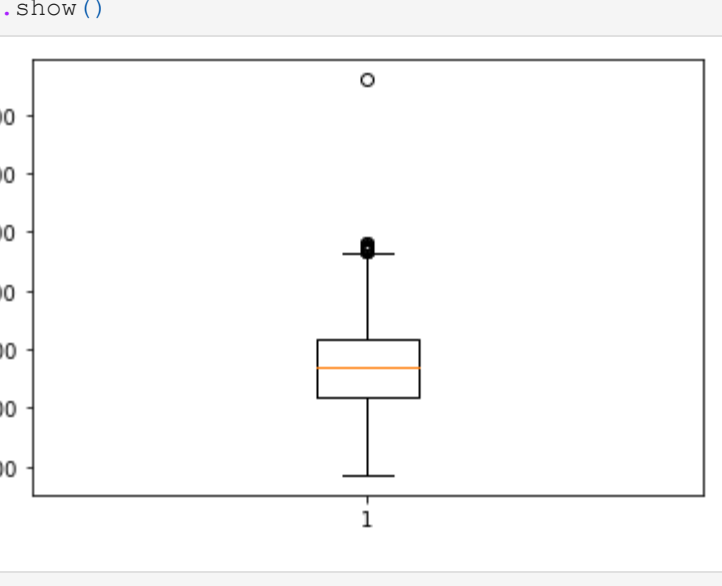
```
In [152]: Q1 = data_prosek['MMRCurrentRetailAveragePrice'].quantile(0.25)
Q3 = data_prosek['MMRCurrentRetailAveragePrice'].quantile(0.75)

# calculate the IQR
IQR = Q3 - Q1

# filter the dataset with the IQR
IQR_outliers = data_prosek['MMRCurrentRetailAveragePrice'][((data_prosek['MMRCurrentRetailAveragePrice'] < (Q1
IQR_outliers
```

```
Out[152]:
RefId
43      20551.0
118         0.0
183         0.0
260         0.0
556         0.0
         ...
6296    17444.0
6607    17488.0
6627        0.0
6684        0.0
6797    17619.0
Name: MMRCurrentRetailAveragePrice, Length: 71, dtype: float64
```

```
In [153]: data_prosek.loc[data_prosek['MMRCurrentRetailAveragePrice'] > (Q3 + 1.5 * IQR), 'MMRCurrentRetailAveragePrice']
```

```
In [154]: data_prosek.loc[data_prosek['MMRCurrentRetailAveragePrice'] < (Q1 - 1.5 * IQR), 'MMRCurrentRetailAveragePrice']
```

```
In [155]: plt.boxplot(data_prosek['VehBCost'])
plt.show()
```

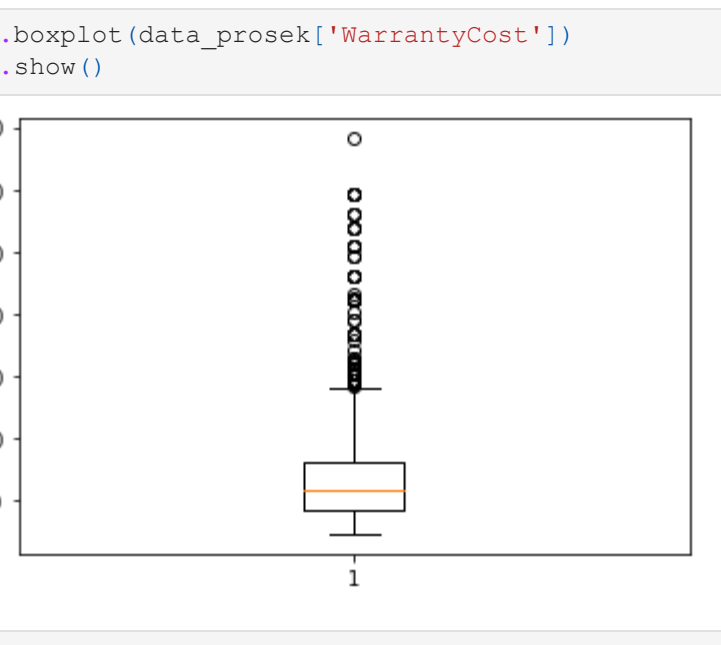


```
In [156]: Q1 = data_prosek['VehBCost'].quantile(0.25)
Q3 = data_prosek['VehBCost'].quantile(0.75)

# calculate the IQR
IQR = Q3 - Q1

# filter the dataset with the IQR
IQR_outliers = data_prosek['VehBCost'][((data_prosek['VehBCost'] < (Q1 - 1.5 * IQR)) |(data_prosek['VehBCost'] 
IQR_outliers
```

```
Out[156]:
RefId
69      11740.00
230     12030.00
256     11900.00
494     11890.00
736     12000.00
1417    11720.00
1784    11920.00
3070    11727.64
3506    11880.00
3582    11705.00
3912    11700.00
4892    11805.00
4894    19000.00
Name: VehBCost, dtype: float64
```

```
In [157]: data_prosek.loc[data_prosek['VehBCost'] > (Q3 + 1.5 * IQR), 'VehBCost'] = (Q3 + 1.5 * IQR)
```

```
In [158]: plt.boxplot(data_prosek['WarrantyCost'])
plt.show()
```



```
In [159]: Q1 = data_prosek['WarrantyCost'].quantile(0.25)
Q3 = data_prosek['WarrantyCost'].quantile(0.75)

# calculate the IQR
IQR = Q3 - Q1

# filter the dataset with the IQR
IQR_outliers = data_prosek['WarrantyCost'][((data_prosek['WarrantyCost'] < (Q1 - 1.5 * IQR)) |(data_prosek['War
IQR_outliers
```

```
Out[159]:
RefId
66      3265
155     4210
349     4922
577     2871
633     3035
        ...
6413    5913
6575    2939
6685    5092
6741    4922
6745    4622
Name: WarrantyCost, Length: 78, dtype: int64
```

```
In [160]: data_prosek.loc[data_prosek['WarrantyCost'] > (Q3 + 1.5 * IQR), 'WarrantyCost'] = (Q3 + 1.5 * IQR)
```

```
In [161]: ####### Normalizacija ########
```

```
In [162]: data_prosek.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 6797 entries, 1 to 6798
Data columns (total 71 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   PurchYear                         6797 non-null   int64
 1   VehicleAge                        6797 non-null   int64
 2   Transmission                      6797 non-null   int64
 3   WheelType                         6797 non-null   int64
 4   VehOdo                            6797 non-null   int64
 5   MMRAcquisitionAuctionAveragePrice 6797 non-null   float64
 6   MMRAcquisitionRetailAveragePrice  6797 non-null   float64
 7   MMRCurrentAuctionAveragePrice     6797 non-null   float64
 8   MMRCurrentRetailAveragePrice      6797 non-null   float64
 9   VehBCost                          6797 non-null   float64
 10  WarrantyCost                      6797 non-null   float64
 11  IsBadBuy                          6797 non-null   object
 12  Make_ACURA                        6797 non-null   uint8
 13  Make_BUICK                        6797 non-null   uint8
 14  Make_CADILLAC                     6797 non-null   uint8
 15  Make_CHEVROLET                    6797 non-null   uint8
 16  Make_CHRYSLER                     6797 non-null   uint8
 17  Make_DODGE                        6797 non-null   uint8
 18  Make_FORD                         6797 non-null   uint8
 19  Make_GMC                          6797 non-null   uint8
 20  Make_HONDA                        6797 non-null   uint8
 21  Make_HUMMER                       6797 non-null   uint8
 22  Make_HYUNDAI                      6797 non-null   uint8
 23  Make_INFINITI                     6797 non-null   uint8
 24  Make_ISUZU                        6797 non-null   uint8
 25  Make_JEEP                         6797 non-null   uint8
 26  Make_KIA                          6797 non-null   uint8
 27  Make_LEXUS                        6797 non-null   uint8
 28  Make_LINCOLN                      6797 non-null   uint8
 29  Make_MAZDA                        6797 non-null   uint8
 30  Make_MERCURY                      6797 non-null   uint8
 31  Make_MINI                         6797 non-null   uint8
 32  Make_MITSUBISHI                   6797 non-null   uint8
 33  Make_NISSAN                       6797 non-null   uint8
 34  Make_OLDSMOBILE                   6797 non-null   uint8
 35  Make_PLYMOUTH                     6797 non-null   uint8
 36  Make_PONTIAC                      6797 non-null   uint8
 37  Make_SATURN                       6797 non-null   uint8
 38  Make_SCION                        6797 non-null   uint8
 39  Make_SUBARU                       6797 non-null   uint8
 40  Make_SUZUKI                       6797 non-null   uint8
 41  Make_TOYOTA                       6797 non-null   uint8
 42  Make_VOLKSWAGEN                   6797 non-null   uint8
 43  Make_VOLVO                        6797 non-null   uint8
 44  Color_BEIGE                       6797 non-null   uint8
 45  Color_BLACK                       6797 non-null   uint8
 46  Color_BLUE                        6797 non-null   uint8
 47  Color_BROWN                       6797 non-null   uint8
 48  Color_GOLD                        6797 non-null   uint8
 49  Color_GREEN                       6797 non-null   uint8
 50  Color_GREY                        6797 non-null   uint8
 51  Color_MAROON                      6797 non-null   uint8
 52  Color_ORANGE                      6797 non-null   uint8
 53  Color_OTHER                       6797 non-null   uint8
 54  Color_PURPLE                      6797 non-null   uint8
 55  Color_RED                         6797 non-null   uint8
 56  Color_SILVER                      6797 non-null   uint8
 57  Color_WHITE                       6797 non-null   uint8
 58  Color_YELLOW                      6797 non-null   uint8
 59  Size_COMPACT                      6797 non-null   uint8
 60  Size_CROSSOVER                    6797 non-null   uint8
 61  Size_LARGE                        6797 non-null   uint8
 62  Size_LARGE SUV                    6797 non-null   uint8
 63  Size_LARGE TRUCK                  6797 non-null   uint8
 64  Size_MEDIUM                       6797 non-null   uint8
 65  Size_MEDIUM SUV                   6797 non-null   uint8
 66  Size_SMALL SUV                    6797 non-null   uint8
 67  Size_SMALL TRUCK                  6797 non-null   uint8
 68  Size_SPECIALTY                    6797 non-null   uint8
 69  Size_SPORTS                       6797 non-null   uint8
 70  Size_VAN                          6797 non-null   uint8
dtypes: float64(6), int64(5), object(1), uint8(59)
memory usage: 1.1+ MB
```

```
In [163]: kol = data_prosek.columns
```

```
In [164]: kol
```

```
Out[164]: Index(['PurchYear', 'VehicleAge', 'Transmission', 'WheelType', 'VehOdo',
       'MMRAcquisitionAuctionAveragePrice', 'MMRAcquisitionRetailAveragePrice',
       'MMRCurrentAuctionAveragePrice', 'MMRCurrentRetailAveragePrice',
       'VehBCost', 'WarrantyCost', 'IsBadBuy', 'Make_ACURA', 'Make_BUICK',
       'Make_CADILLAC', 'Make_CHEVROLET', 'Make_CHRYSLER', 'Make_DODGE',
       'Make_FORD', 'Make_GMC', 'Make_HONDA', 'Make_HUMMER', 'Make_HYUNDAI',
       'Make_INFINITI', 'Make_ISUZU', 'Make_JEEP', 'Make_KIA', 'Make_LEXUS',
       'Make_LINCOLN', 'Make_MAZDA', 'Make_MERCURY', 'Make_MINI',
       'Make_MITSUBISHI', 'Make_NISSAN', 'Make_OLDSMOBILE', 'Make_PLYMOUTH',
       'Make_PONTIAC', 'Make_SATURN', 'Make_SCION', 'Make_SUBARU',
       'Make_SUZUKI', 'Make_TOYOTA', 'Make_VOLKSWAGEN', 'Make_VOLVO',
       'Color_BEIGE', 'Color_BLACK', 'Color_BLUE', 'Color_BROWN', 'Color_GOLD',
       'Color_GREEN', 'Color_GREY', 'Color_MAROON', 'Color_ORANGE',
       'Color_OTHER', 'Color_PURPLE', 'Color_RED', 'Color_SILVER',
       'Color_WHITE', 'Color_YELLOW', 'Size_COMPACT', 'Size_CROSSOVER',
       'Size_LARGE', 'Size_LARGE SUV', 'Size_LARGE TRUCK', 'Size_MEDIUM',
       'Size_MEDIUM SUV', 'Size_SMALL SUV', 'Size_SMALL TRUCK',
       'Size_SPECIALTY', 'Size_SPORTS', 'Size_VAN'],
      dtype='object')
```

```
In [165]: from sklearn.preprocessing import MinMaxScaler
r = MinMaxScaler(feature_range=(0, 1))
data_prosek_norm = r.fit_transform(data_prosek)
data_prosek_norm = pd.DataFrame(data_prosek_norm, columns = kol)
data_prosek_norm.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6797 entries, 0 to 6796
Data columns (total 71 columns):
 #   Column                            Non-Null Count  Dtype
---  ------                            --------------  -----
 0   PurchYear                         6797 non-null   float64
 1   VehicleAge                        6797 non-null   float64
 2   Transmission                      6797 non-null   float64
 3   WheelType                         6797 non-null   float64
 4   VehOdo                            6797 non-null   float64
 5   MMRAcquisitionAuctionAveragePrice 6797 non-null   float64
 6   MMRAcquisitionRetailAveragePrice  6797 non-null   float64
 7   MMRCurrentAuctionAveragePrice     6797 non-null   float64
 8   MMRCurrentRetailAveragePrice      6797 non-null   float64
 9   VehBCost                          6797 non-null   float64
 10  WarrantyCost                      6797 non-null   float64
 11  IsBadBuy                          6797 non-null   float64
 12  Make_ACURA                        6797 non-null   float64
 13  Make_BUICK                        6797 non-null   float64
 14  Make_CADILLAC                     6797 non-null   float64
 15  Make_CHEVROLET                    6797 non-null   float64
 16  Make_CHRYSLER                     6797 non-null   float64
 17  Make_DODGE                        6797 non-null   float64
 18  Make_FORD                         6797 non-null   float64
 19  Make_GMC                          6797 non-null   float64
 20  Make_HONDA                        6797 non-null   float64
 21  Make_HUMMER                       6797 non-null   float64
 22  Make_HYUNDAI                      6797 non-null   float64
 23  Make_INFINITI                     6797 non-null   float64
 24  Make_ISUZU                        6797 non-null   float64
 25  Make_JEEP                         6797 non-null   float64
 26  Make_KIA                          6797 non-null   float64
 27  Make_LEXUS                        6797 non-null   float64
 28  Make_LINCOLN                      6797 non-null   float64
 29  Make_MAZDA                        6797 non-null   float64
 30  Make_MERCURY                      6797 non-null   float64
 31  Make_MINI                         6797 non-null   float64
 32  Make_MITSUBISHI                   6797 non-null   float64
 33  Make_NISSAN                       6797 non-null   float64
 34  Make_OLDSMOBILE                   6797 non-null   float64
 35  Make_PLYMOUTH                     6797 non-null   float64
 36  Make_PONTIAC                      6797 non-null   float64
 37  Make_SATURN                       6797 non-null   float64
 38  Make_SCION                        6797 non-null   float64
 39  Make_SUBARU                       6797 non-null   float64
 40  Make_SUZUKI                       6797 non-null   float64
 41  Make_TOYOTA                       6797 non-null   float64
 42  Make_VOLKSWAGEN                   6797 non-null   float64
 43  Make_VOLVO                        6797 non-null   float64
 44  Color_BEIGE                       6797 non-null   float64
 45  Color_BLACK                       6797 non-null   float64
 46  Color_BLUE                        6797 non-null   float64
 47  Color_BROWN                       6797 non-null   float64
 48  Color_GOLD                        6797 non-null   float64
 49  Color_GREEN                       6797 non-null   float64
 50  Color_GREY                        6797 non-null   float64
 51  Color_MAROON                      6797 non-null   float64
 52  Color_ORANGE                      6797 non-null   float64
 53  Color_OTHER                       6797 non-null   float64
 54  Color_PURPLE                      6797 non-null   float64
 55  Color_RED                         6797 non-null   float64
 56  Color_SILVER                      6797 non-null   float64
 57  Color_WHITE                       6797 non-null   float64
 58  Color_YELLOW                      6797 non-null   float64
 59  Size_COMPACT                      6797 non-null   float64
 60  Size_CROSSOVER                    6797 non-null   float64
 61  Size_LARGE                        6797 non-null   float64
 62  Size_LARGE SUV                    6797 non-null   float64
 63  Size_LARGE TRUCK                  6797 non-null   float64
 64  Size_MEDIUM                       6797 non-null   float64
 65  Size_MEDIUM SUV                   6797 non-null   float64
 66  Size_SMALL SUV                    6797 non-null   float64
 67  Size_SMALL TRUCK                  6797 non-null   float64
 68  Size_SPECIALTY                    6797 non-null   float64
 69  Size_SPORTS                       6797 non-null   float64
 70  Size_VAN                          6797 non-null   float64
dtypes: float64(71)
memory usage: 3.7 MB
```

```
In [166]: data_prosek_norm
```

Out[166]:

	PurchYear	VehicleAge	Transmission	WheelType	VehOdo	MMRAcquisitionAuctionAveragePrice	MMRAcquisitionRetailAveragePrice	MMI
0	0.0	0.428571	1.0	0.0	0.527277	0.245825	0.405703	
1	0.0	0.571429	1.0	1.0	0.425985	0.145330	0.272183	
2	0.0	0.428571	0.0	0.0	0.472364	0.300411	0.451281	
3	0.0	0.571429	1.0	0.0	0.616907	0.299489	0.391854	
4	0.0	0.571429	1.0	1.0	0.556539	0.491881	0.606422	
...	...	...	...	...	...	...	...	
6792	0.0	0.285714	1.0	0.0	0.363305	0.299797	0.275630	
6793	0.0	0.428571	1.0	1.0	0.670373	0.545392	0.477518	
6794	0.0	0.857143	1.0	1.0	0.550690	0.391156	0.350776	
6795	0.0	0.285714	1.0	0.0	0.479463	0.936240	0.798843	
6796	0.0	0.857143	1.0	1.0	0.773854	0.421097	0.375376	

6797 rows × 71 columns

Klasifikacija

```
In [168]: # Koristicemo sledece algoritme klasifikacije:

#Naivni Bajes
#Stablo odlucivanja
#K najblizih suseda (KNN)
#Logisticku regresiju
#Ansambl algoritme
#Ovim algoritmima cemo analizirati podatke nad originalnim skupom podataka i nad podacima
#dobijenim koriscenjem razlicitih metoda za selekciju atributa, a sve u svrhu odabira najboljeg modela za nas k
```

```
In [169]: alg = ['Naivni Bajes', 'KNN', 'Stablo odlucivanja', 'Logisticka regresija', 'Voting', 'Bagging', 'Random forest
data_alg_rez = pd.DataFrame(alg)
```

```
In [170]: data_alg_rez
```

Out[170]:

	0
0	Naivni Bajes
1	KNN
2	Stablo odlucivanja
3	Logisticka regresija
4	Voting
5	Bagging
6	Random forest

```
In [171]: data_alg_rez.columns = ['Algoritam']
data_alg_rez = data_alg_rez.set_index('Algoritam')
```

```
In [172]: x_n = data_prosek_norm.loc[:, data_prosek_norm.columns != 'IsBadBuy']
y_n = data_prosek_norm.loc[:, 'IsBadBuy']
```

```
In [173]: # Klasifikacija je tip nadgledanog ucenja, pa moramo podeliti dataset na trening i test
```

```
In [174]: from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(x_n, y_n, test_size = 0.3, random_state=10)
```

```
In [175]: model_nb = GaussianNB()
model_knn = KNeighborsClassifier()
model_dt = DecisionTreeClassifier()
model_log = LogisticRegression()
model_voting = VotingClassifier(voting='soft', estimators=[('nb',model_nb),('knn',model_knn),('tree', model_dt)
model_bagging = BaggingClassifier(base_estimator=model_log, bootstrap=10)
model_rf = RandomForestClassifier(n_estimators=100, oob_score=True)

lista = list()
for model in [model_nb, model_knn, model_dt, model_log, model_voting, model_bagging, model_rf]:
    scores = cross_val_score(model, X_train, Y_train, cv=10, scoring='roc_auc')
    lista.append(round(np.mean(scores)*100, 2))
data_alg_rez['Inicijalni skup podataka'] = lista
```

```
In [176]: # Odredjujemo optimalan broj modela logisticke regresije u Bagging-u
```

```
In [177]: model_bagging_proba20 = BaggingClassifier(base_estimator=model_log, bootstrap=20)
model_bagging_proba40 = BaggingClassifier(base_estimator=model_log, bootstrap=40)
model_bagging_proba10 = BaggingClassifier(base_estimator=model_log, bootstrap=10)
```

```
In [178]: print(cross_val_score(model_bagging_proba10, X_train, Y_train, cv=10, scoring='roc_auc').mean())
print(cross_val_score(model_bagging_proba20, X_train, Y_train, cv=10, scoring='roc_auc').mean())
print(cross_val_score(model_bagging_proba40, X_train, Y_train, cv=10, scoring='roc_auc').mean())

0.6697826611745151
0.6694423882271782
0.6711623686477914
```

```
In [179]: # Vidimo da izmenom parametra bootstrap (koliko ce modela uzimati u obzir) ne dobijamo bolje rezultate, tako da
# 10 jer je to najmanji broj, a samim tim uticemo i na vreme izvrsavanja
```

```
In [180]: data_alg_rez
```

Out[180]:

	Inicijalni skup podataka
Algoritam	
Naivni Bajes	50.34
KNN	55.23
Stablo odlucivanja	51.51
Logisticka regresija	67.14
Voting	61.01
Bagging	66.96
Random forest	62.77

```
In [181]: data_original = pd.DataFrame()
```

```
In [182]: data_original = data.loc[:, ['MMRAcquisitionAuctionCleanPrice', 'MMRAcquisitonRetailCleanPrice', 'MMRCurrentAuc
```

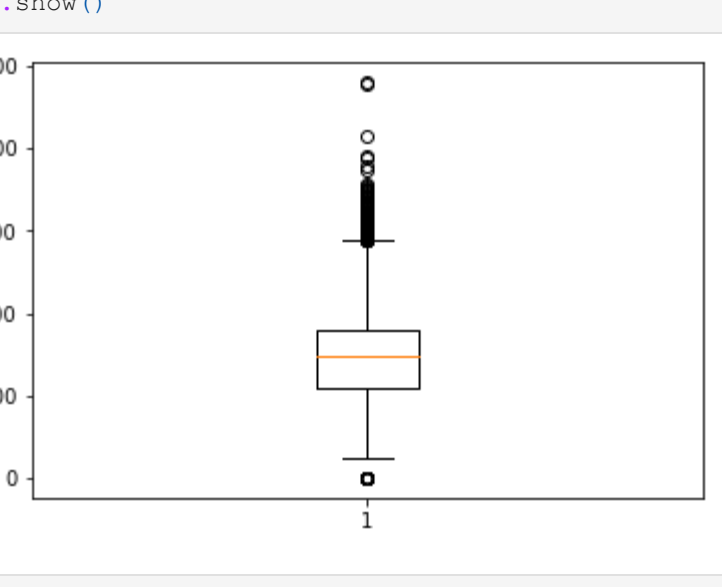
```
In [183]: data_original
```

Out[183]:

	MMRAcquisitionAuctionCleanPrice	MMRAcquisitonRetailCleanPrice	MMRCurrentAuctionCleanPrice	MMRCurrentRetailCleanPrice
RefId				
1	4760.0	8457.0	5557.0	8702.0
2	2675.0	5690.0	2646.0	5518.0
3	5054.0	8707.0	4384.0	7911.0
4	4908.0	8577.0	5827.0	9451.0
5	8051.0	12443.0	8934.0	12560.0
...	...	...	...	...
6794	4892.0	5783.0	4714.0	8766.0
6795	8601.0	9789.0	8859.0	12190.0
6796	5988.0	6967.0	7185.0	9754.0
6797	13568.0	15153.0	15015.0	19161.0
6798	6823.0	7869.0	7686.0	10173.0

6797 rows × 4 columns

```
In [184]: plt.boxplot(data_original['MMRAcquisitionAuctionCleanPrice'])
plt.show()
```



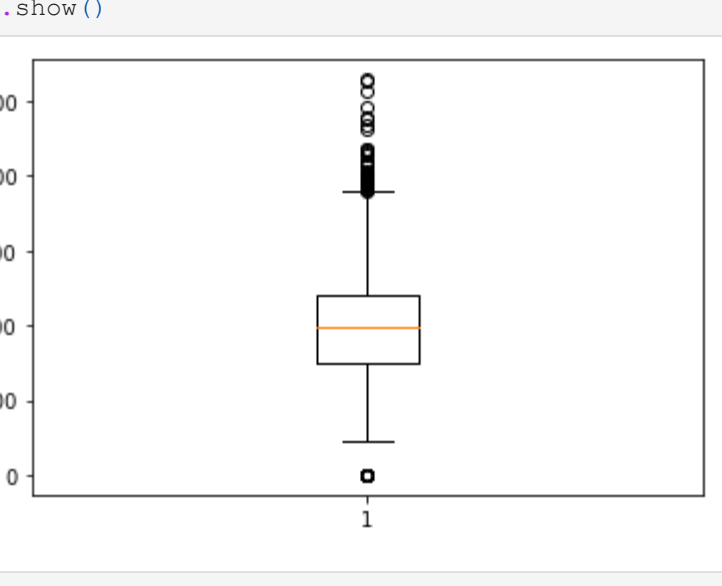
```
In [185]: Q1 = data_original['MMRAcquisitionAuctionCleanPrice'].quantile(0.25)
Q3 = data_original['MMRAcquisitionAuctionCleanPrice'].quantile(0.75)

# calculate the IQR
IQR = Q3 - Q1

# filter the dataset with the IQR
IQR_outliers = data_original['MMRAcquisitionAuctionCleanPrice'][((data_original['MMRAcquisitionAuctionCleanPric
IQR_outliers
```

```
Out[185]:
RefId
9       15664.0
43      18927.0
118         0.0
183         0.0
216     15016.0
         ...
6353    16455.0
6472    14518.0
6607    15072.0
6627        1.0
6684        0.0
Name: MMRAcquisitionAuctionCleanPrice, Length: 164, dtype: float64
```

```
In [186]: data_original.loc[data_original['MMRAcquisitionAuctionCleanPrice'] > (Q3 + 1.5 * IQR), 'MMRAcquisitionAuctionCl
```

```
In [187]: data_original.loc[data_original['MMRAcquisitionAuctionCleanPrice']  < (Q1 - 1.5 * IQR), 'MMRAcquisitionAuctionC
```

```
In [188]: plt.boxplot(data_original['MMRAcquisitonRetailCleanPrice'])
plt.show()
```



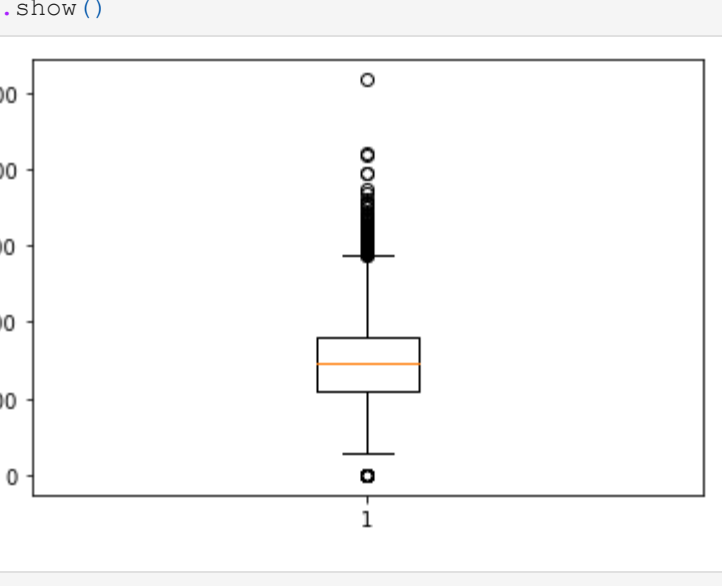
```
In [189]: Q1 = data_original['MMRAcquisitonRetailCleanPrice'].quantile(0.25)
Q3 = data_original['MMRAcquisitonRetailCleanPrice'].quantile(0.75)

# calculate the IQR
IQR = Q3 - Q1

# filter the dataset with the IQR
IQR_outliers = data_original['MMRAcquisitonRetailCleanPrice'][((data_original['MMRAcquisitonRetailCleanPrice'] 
IQR_outliers
```

```
Out[189]:
RefId
9       21702.0
43      23457.0
118         0.0
183         0.0
230     19320.0
         ...
6224    19721.0
6353    20768.0
6607    19412.0
6627        0.0
6684        0.0
Name: MMRAcquisitonRetailCleanPrice, Length: 113, dtype: float64
```

```
In [190]: data_original.loc[data_original['MMRAcquisitonRetailCleanPrice'] > (Q3 + 1.5 * IQR), 'MMRAcquisitonRetailCleanP
```

```
In [191]: data_original.loc[data_original['MMRAcquisitonRetailCleanPrice']  < (Q1 - 1.5 * IQR), 'MMRAcquisitonRetailClean
```

```
In [192]: plt.boxplot(data_original['MMRCurrentAuctionCleanPrice'])
plt.show()
```



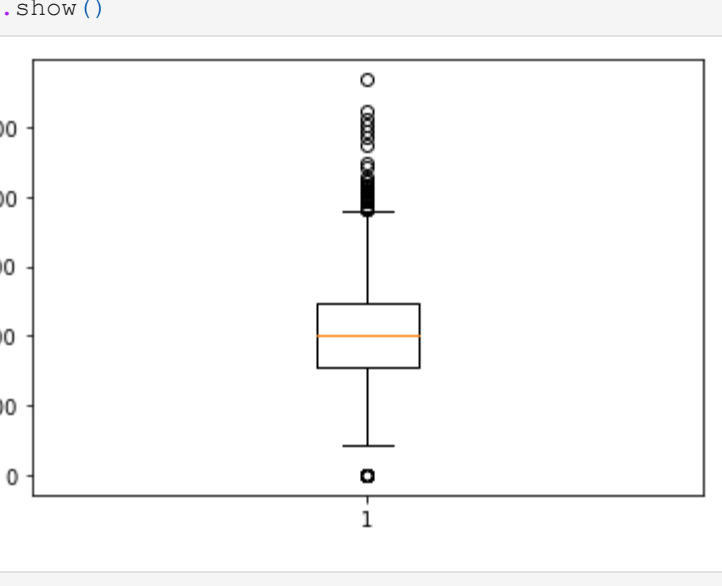
```
In [193]: Q1 = data_original['MMRCurrentAuctionCleanPrice'].quantile(0.25)
Q3 = data_original['MMRCurrentAuctionCleanPrice'].quantile(0.75)

# calculate the IQR
IQR = Q3 - Q1

# filter the dataset with the IQR
IQR_outliers = data_original['MMRCurrentAuctionCleanPrice'][((data_original['MMRCurrentAuctionCleanPrice'] < (Q
IQR_outliers
```

```
Out[193]:
RefId
9       14974.0
43      18461.0
118         0.0
183         0.0
216     14858.0
         ...
6472    16108.0
6607    14908.0
6627        1.0
6684        0.0
6797    15015.0
Name: MMRCurrentAuctionCleanPrice, Length: 132, dtype: float64
```

```
In [194]: data_original.loc[data_original['MMRCurrentAuctionCleanPrice'] > (Q3 + 1.5 * IQR), 'MMRCurrentAuctionCleanPrice
```

```
In [195]: data_original.loc[data_original['MMRCurrentAuctionCleanPrice']  < (Q1 - 1.5 * IQR), 'MMRCurrentAuctionCleanPric
```

```
In [196]: plt.boxplot(data_original['MMRCurrentRetailCleanPrice'])
plt.show()
```



```
In [197]: Q1 = data_original['MMRCurrentRetailCleanPrice'].quantile(0.25)
Q3 = data_original['MMRCurrentRetailCleanPrice'].quantile(0.75)

# calculate the IQR
IQR = Q3 - Q1

# filter the dataset with the IQR
IQR_outliers = data_original['MMRCurrentRetailCleanPrice'][((data_original['MMRCurrentRetailCleanPrice'] < (Q1
IQR_outliers
```

```
Out[197]:
RefId
9       19975.0
43      22416.0
118         0.0
183         0.0
260         0.0
         ...
6296    19616.0
6310    19096.0
6627        0.0
6684        0.0
6797    19161.0
Name: MMRCurrentRetailCleanPrice, Length: 80, dtype: float64
```

```
In [198]: data_original.loc[data_original['MMRCurrentRetailCleanPrice'] > (Q3 + 1.5 * IQR), 'MMRCurrentRetailCleanPrice']
```

```
In [199]: data_original.loc[data_original['MMRCurrentRetailCleanPrice']  < (Q1 - 1.5 * IQR), 'MMRCurrentRetailCleanPrice'
```

```
In [200]: kol1 = data_original.columns
r = MinMaxScaler(feature_range=(0, 1))
data_original = r.fit_transform(data_original)
data_original = pd.DataFrame(data_original, columns = kol1)
data_original.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6797 entries, 0 to 6796
Data columns (total 4 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   MMRAcquisitionAuctionCleanPrice  6797 non-null   float64
 1   MMRAcquisitonRetailCleanPrice    6797 non-null   float64
 2   MMRCurrentAuctionCleanPrice      6797 non-null   float64
 3   MMRCurrentRetailCleanPrice       6797 non-null   float64
dtypes: float64(4)
memory usage: 212.5 KB
```

```
In [201]: data_original = pd.concat([data_original, data_prosek_norm], axis = 1)
```

```
In [202]: data_original.head(50)
```

Out[202]:

	MMRAcquisitionAuctionCleanPrice	MMRAcquisitonRetailCleanPrice	MMRCurrentAuctionCleanPrice	MMRCurrentRetailCleanPrice	PurchYear
0	0.329501	0.427505	0.383410	0.424365	0.0
1	0.184669	0.276797	0.179387	0.246963	0.0
2	0.349924	0.441122	0.301198	0.380293	0.0
3	0.339782	0.434041	0.402334	0.466097	0.0
4	0.558106	0.644608	0.620094	0.639319	0.0
5	0.406606	0.471405	0.438008	0.492172	1.0
6	0.326723	0.404630	0.370514	0.442640	1.0
7	0.558593	0.625436	0.512581	0.602379	1.0
8	1.000000	1.000000	1.000000	1.000000	1.0
9	0.589157	0.641667	0.650652	0.680939	1.0
10	0.739059	0.766503	0.807717	0.809589	1.0
11	0.197659	0.381481	0.217725	0.249248	1.0
12	0.386600	0.484314	0.387896	0.480527	1.0
13	0.525111	0.550763	0.557296	0.569896	1.0
14	0.392435	0.460948	0.418734	0.458352	1.0
15	0.383544	0.441449	0.328602	0.399627	1.0
16	0.453633	0.481046	0.440181	0.495069	1.0
17	0.668137	0.668736	0.659202	0.676092	1.0
18	0.732599	0.727505	0.674271	0.689464	1.0
19	0.518234	0.562200	0.521832	0.550507	1.0
20	0.425222	0.480610	0.410043	0.453950	1.0
21	0.385628	0.417919	0.347456	0.428209	1.0
22	0.366942	0.412582	0.358880	0.407483	1.0
23	0.637503	0.711002	0.622407	0.637202	1.0
24	0.260593	0.288508	0.245059	0.257605	1.0
25	0.355759	0.429194	0.337574	0.425925	1.0
26	0.310399	0.371024	0.324117	0.383079	1.0
27	0.420429	0.487418	0.400161	0.462698	1.0
28	0.648896	0.661057	0.613506	0.645114	1.0
29	0.398131	0.484150	0.368552	0.398735	1.0
30	0.336448	0.440359	0.279542	0.323852	1.0
31	0.594575	0.671514	0.612454	0.629624	1.0
32	0.211760	0.286438	0.256413	0.290199	1.0
33	0.522958	0.530991	0.468005	0.497242	1.0
34	0.610065	0.668137	0.597946	0.612241	1.0
35	0.463497	0.537146	0.488821	0.528276	1.0
36	0.388337	0.428922	0.402895	0.431942	1.0
37	0.796020	0.824129	0.829584	0.825524	1.0
38	0.497326	0.580773	0.549236	0.577697	1.0
39	0.367359	0.445806	0.422449	0.453505	1.0
40	0.518234	0.580501	0.554633	0.563768	1.0
41	0.310816	0.410022	0.322014	0.353828	1.0
42	1.000000	1.000000	1.000000	1.000000	1.0
43	0.705786	0.592756	0.646797	0.527886	0.0
44	0.786989	0.661547	0.826710	0.682388	0.0
45	0.337073	0.280556	0.318860	0.246351	0.0
46	0.566512	0.474837	0.591148	0.480137	0.0
47	0.460301	0.384858	0.465763	0.372493	0.0
48	0.460301	0.384858	0.470108	0.376226	0.0
49	0.386323	0.322222	0.436817	0.347643	0.0

50 rows × 75 columns

```
In [203]: data_original.loc[:,'IsBadBuy'] = data_original.loc[:,'IsBadBuy'] - 1
```

In [204…
```python
data_original.loc[:,'IsBadBuy'] = data_original.loc[:,'IsBadBuy'] * (-1)
```

In [205…
```python
x_o_n = data_original.loc[:, data_original.columns != 'IsBadBuy']
y_o_n = data_original.loc[:, 'IsBadBuy']
```

In [206…
```python
# Moramo i drugi dataset sa svim MMR podacima da podelimo na trening i test
```

In [207…
```python
from sklearn.model_selection import train_test_split
X_train_o, X_test_o, Y_train_o, Y_test_o = train_test_split(x_o_n, y_o_n, test_size = 0.3, random_state=10)
```

In [208…
```python
lista1 = list()
for model in [model_nb, model_knn, model_dt, model_log, model_voting, model_bagging, model_rf]:
    scores = cross_val_score(model, X_train_o, Y_train_o, cv=10, scoring='roc_auc')
    lista1.append(round(np.mean(scores)*100, 2))
data_alg_rez['Potpuni skup podataka'] = lista1
```

In [209…
```python
data_alg_rez.sort_values
```

Out[209]:
```
<bound method DataFrame.sort_values of                          Inicijalni skup podataka  Potpuni skup podataka
Algoritam
Naivni Bajes                            50.34                  59.91
KNN                                     55.23                  55.11
Stablo odlucivanja                      51.51                  52.55
Logisticka regresija                    67.14                  67.20
Voting                                  61.01                  61.73
Bagging                                 66.96                  67.03
Random forest                           62.77                  63.36>
```

In [210…
```python
# Dalje cemo raditi sa potpunim skupom podataka, jer vidimo da nema znacajne razlike u rezultatima
```

In [276…
```python
lista2 = list()
for model in [model_nb, model_knn, model_dt, model_log, model_voting, model_bagging, model_rf]:
    scores = cross_val_score(model, X_train_o, Y_train_o, cv=10, scoring='accuracy')
    lista2.append(round(np.mean(scores)*100, 2))
data_alg_rez['Potpuni skup podataka sa ACC'] = lista2
```

In [277…
```python
data_alg_rez
```

Out[277]:

Algoritam	Inicijalni skup podataka	Potpuni skup podataka	Potpuni skup podataka sa ACC
Naivni Bajes	50.34	59.91	14.36
KNN	55.23	55.11	85.68
Stablo odlucivanja	51.51	52.55	78.05
Logisticka regresija	67.14	67.20	87.16
Voting	61.01	61.73	87.18
Bagging	66.96	67.03	87.16
Random forest	62.77	63.36	87.11

In [213…
```python
model_voting2 = VotingClassifier(voting='soft', estimators=[('knn',model_knn),('tree', model_dt), ('log', model
```

In [214…
```python
cross_val_score(model_voting2, x_o_n, y_o_n, cv=10, scoring='accuracy').mean()
```

Out[214]:
```
0.8540561379190853
```

In [215…
```python
# Vidimo da nam Voting radi bolje ukoliko izbacimo Naivnog Bajesa jer on pretpostavlja da su promenljive medjus
# a znamo da imamo 8 varijabli koje su bas jako korelisane
```

In [218…
```python
from mlxtend.classifier import StackingClassifier
```

In [219…
```python
model_stacking = StackingClassifier(classifiers=[model_knn, model_dt, model_log], meta_classifier=LogisticRegre
```

In [220…
```python
cross_val_score(model_stacking, x_o_n, y_o_n, cv=10, scoring='roc_auc').mean()
```

Out[220]:
```
0.6412432022624308
```

In [221…
```python
cross_val_score(model_stacking, x_o_n, y_o_n, cv=10, scoring='accuracy').mean()
```

Out[221]:
```
0.7785792255046349
```

In [222…
```python
# Vidimo da nam najbolje rezultate daju logisticka regresija i KNN, kao i kombinacija  ova dva algoritma i drve
# koristeci Voting i Stacking, ali cemo Voting izbaciti iz dalje analize i ostalim modelima uraditi predvidjanj
# kako bismo poredile rezultate i odlucile se za najbolji model
```

Predvidjanja

In [224…
```python
model_log.fit(X_train_o, Y_train_o)
predictions_log = model_log.predict(X_train_o)
pd.DataFrame({'stvarno':Y_train_o, 'predvidjeno':predictions_log}).head(13)
```

Out[224]:

	stvarno	predvidjeno
554	1.0	1.0
5591	1.0	1.0
2516	1.0	1.0
3588	-0.0	1.0
3760	1.0	1.0
53	1.0	1.0
1775	1.0	1.0
2950	1.0	1.0
4738	1.0	1.0
3012	1.0	1.0
2137	1.0	1.0
3226	1.0	1.0
3253	1.0	1.0

In [315…
```python
model_knn.fit(X_train_o, Y_train_o)
predictions_knn = model_knn.predict(X_train_o)
pd.DataFrame({'stvarno':Y_train_o, 'predvidjeno':predictions_knn}).head(10)
```

Out[315]:

	stvarno	predvidjeno
554	1.0	1.0
5591	1.0	1.0
2516	1.0	1.0
3588	-0.0	1.0
3760	1.0	1.0
53	1.0	1.0
1775	1.0	1.0
2950	1.0	1.0
4738	1.0	1.0
3012	1.0	1.0

In [226…
```python
model_stacking.fit(X_train_o, Y_train_o)
predictions_stacking = model_stacking.predict(X_train_o)
pd.DataFrame({'stvarno':Y_train_o, 'predvidjeno':predictions_stacking}).head(13)
```

Out[226]:

	stvarno	predvidjeno
554	1.0	1.0
5591	1.0	1.0
2516	1.0	1.0
3588	-0.0	-0.0
3760	1.0	1.0
53	1.0	1.0
1775	1.0	1.0
2950	1.0	1.0
4738	1.0	1.0
3012	1.0	1.0
2137	1.0	1.0
3226	1.0	1.0
3253	1.0	1.0

In [227…
```python
from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, classification_report, con
```

In [228…
```python
print('Recall logisticka regresija: ', recall_score(Y_test_o, model_log.predict(X_test_o)))
print('Recall KNN: ', recall_score(Y_test_o, model_knn.predict(X_test_o)))
print('Recall Stacking: ', recall_score(Y_test_o, model_stacking.predict(X_test_o)))

print('Precision logisticka regresija: ', precision_score(Y_test_o, model_log.predict(X_test_o)))
print('Precision KNN: ', precision_score(Y_test_o, model_knn.predict(X_test_o)))
print('Precision Stacking: ', precision_score(Y_test_o, model_stacking.predict(X_test_o)))

print('F1 logisticka regresija: ',f1_score(Y_test_o, model_log.predict(X_test_o)))
print('F1 KNN: ', f1_score(Y_test_o, model_knn.predict(X_test_o)))
print('F1 Stacking: ',f1_score(Y_test_o, model_stacking.predict(X_test_o)))
```

```
Recall logisticka regresija:  1.0
Recall KNN:  0.9854260089686099
Recall Stacking:  0.8806053811659192
Precision logisticka regresija:  0.8745098039215686
Precision KNN:  0.8772455089820359
Precision Stacking:  0.8815937149270483
F1 logisticka regresija:  0.9330543933054394
F1 KNN:  0.928194297782471
F1 Stacking:  0.8810992708917554
```

In [229…
```python
# Menjamo granicu odlucivanja jer nam disbalans klasa izlazne promenljive znatno utice na predvidjanje
```

In [230…
```python
prob_log = model_log.predict_proba(X_test_o)[:, 1] >= 0.9
prob_knn = model_knn.predict_proba(X_test_o)[:, 1] >= 0.9
prob_stacking = model_stacking.predict_proba(X_test_o)[:, 1] >= 0.9
```

In [231…
```python
print('Recall logisticka regresija: ', recall_score(Y_test_o, prob_log))
print('Recall KNN: ', recall_score(Y_test_o, prob_knn))
print('Recall Stacking: ', recall_score(Y_test_o, prob_stacking))

print('Precision logisticka regresija: ', precision_score(Y_test_o, prob_log))
print('Precision KNN: ', precision_score(Y_test_o, prob_knn))
print('Precision Stacking: ', precision_score(Y_test_o, prob_stacking))

print('F1 logisticka regresija: ',f1_score(Y_test_o, prob_log))
print('F1 KNN: ', f1_score(Y_test_o, prob_knn))
print('F1 Stacking: ',f1_score(Y_test_o, prob_stacking))
```

```
Recall logisticka regresija:  0.48878923766816146
Recall KNN:  0.5571748878923767
Recall Stacking:  0.8806053811659192
Precision logisticka regresija:  0.9247083775185578
Precision KNN:  0.9003623188405797
Precision Stacking:  0.8815937149270483
F1 logisticka regresija:  0.6395306197286396
F1 KNN:  0.6883656509695291
F1 Stacking:  0.8810992708917554
```

In [232…
```python
confusion_matrix(Y_test_o, model_log.predict(X_test_o))
```

Out[232]:
```
array([[   0,  256],
       [   0, 1784]], dtype=int64)
```

In [233…
```python
confusion_matrix(Y_test_o, model_knn.predict(X_test_o))
```

Out[233]:
```
array([[  10,  246],
       [  26, 1758]], dtype=int64)
```

In [234…
```python
confusion_matrix(Y_test_o, model_stacking.predict(X_test_o))
```

Out[234]:
```
array([[  45,  211],
       [ 213, 1571]], dtype=int64)
```

In [235…
```python
# Uocavamo da smo dobili zadovoljavajuce rezultate za precision sto nam je svakako bitnije i mozemo videti da n
# regresija definitivno radi najbolje
```

In [236…
```python
# Posto su logisticka regresija i Stacking dali dobre rezultate, sad cemo izvrsiti optimizaciju parametara
# za stablo odlucivanja i KNN kako bismo i njih pokusale da poboljsamo
```

In [284…
```python
data_alg_rez
```

Out[284]:

Algoritam	Inicijalni skup podataka	Potpuni skup podataka	Potpuni skup podataka sa ACC
Naivni Bajes	50.34	59.91	14.36
KNN	55.23	55.11	85.68
Stablo odlucivanja	51.51	52.55	78.05
Logisticka regresija	67.14	67.20	87.16
Voting	61.01	61.73	87.18
Bagging	66.96	67.03	87.16
Random forest	62.77	63.36	87.11

In [285…
```python
selection = VarianceThreshold(0.05)
selection.fit(x_o_n)
x_o_filter = x_o_n.loc[:, selection.get_support()]
```

In [286…
```python
x_o_filter
```

Out[286]:

	PurchYear	VehicleAge	WheelType	Make_CHEVROLET	Make_CHRYSLER	Make_DODGE	Make_FORD	Make_PONTIAC	Color_BLACK	Color
0	0.0	0.428571	0.0	0.0	0.0	1.0	0.0	0.0	0.0	
1	0.0	0.571429	1.0	0.0	0.0	1.0	0.0	0.0	0.0	
2	0.0	0.428571	0.0	0.0	0.0	0.0	1.0	0.0	0.0	
3	0.0	0.571429	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.571429	1.0	0.0	0.0	1.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	
6792	0.0	0.285714	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
6793	0.0	0.428571	1.0	0.0	0.0	0.0	0.0	0.0	0.0	
6794	0.0	0.857143	1.0	1.0	0.0	0.0	0.0	0.0	0.0	
6795	0.0	0.285714	0.0	0.0	0.0	0.0	0.0	0.0	1.0	
6796	0.0	0.857143	1.0	0.0	0.0	1.0	0.0	0.0	0.0	

6797 rows × 20 columns

In [287…
```python
X_train_f, X_test_f, Y_train_f, Y_test_f = train_test_split(x_o_filter, y_o_n, test_size = 0.3, random_state=10
```

In [288…
```python
lista5 = list()
for model in [model_nb, model_knn, model_dt, model_log, model_voting, model_bagging, model_rf]:
    scores = cross_val_score(model, X_train_f, Y_train_f, cv=10, scoring='roc_auc')
    lista5.append(round(np.mean(scores)*100, 2))
data_alg_rez['Potpuni skup podataka filter'] = lista5
```

In [290…
```python
lista6 = list()
for model in [model_nb, model_knn, model_dt, model_log, model_voting, model_bagging, model_rf]:
    scores = cross_val_score(model, X_train_f, Y_train_f, cv=10, scoring='accuracy')
    lista6.append(round(np.mean(scores)*100, 2))
data_alg_rez['Potpuni skup podataka filter ACC'] = lista6
```

In [291…
```python
data_alg_rez
```

Out[291]:

Algoritam	Inicijalni skup podataka	Potpuni skup podataka	Potpuni skup podataka sa ACC	Potpuni skup podataka filter	Potpuni skup podataka filter ACC
Naivni Bajes	50.34	59.91	14.36	61.44	81.46
KNN	55.23	55.11	85.68	56.20	85.31
Stablo odlucivanja	51.51	52.55	78.05	53.12	77.63
Logisticka regresija	67.14	67.20	87.16	65.53	87.18
Voting	61.01	61.73	87.18	63.25	87.18
Bagging	66.96	67.03	87.16	65.45	87.18
Random forest	62.77	63.36	87.11	59.78	83.18

Optimizacija parametara

In [311…
```python
knn_params = {'n_neighbors': [1,2,3,4,5,6,7,8,9,10,12,20,30,50,100]}
grid = GridSearchCV(model_knn, knn_params, cv=10, scoring='roc_auc')
grid.fit(X_train_f,Y_train_f)

print('Best param: ', grid.best_params_)
```

```
Best param:  {'n_neighbors': 100}
```

In [301…
```python
max_depths = {'max_depth': np.linspace(1, 40, 40, endpoint=True)}
```

In [302…
```python
grid = GridSearchCV(model_dt, max_depths, cv=10, scoring='roc_auc')
grid.fit(X_train_f,Y_train_f)

print('Best param: ', grid.best_params_)
```

```
Best param:  {'max_depth': 2.0}
```

In [305…
```python
rez_optimizovano = pd.DataFrame()
rez_optimizovano['Pre optimizacije'] = data_alg_rez.iloc[:,3]
rez_optimizovano
```

Out[305]:

Algoritam	Pre optimizacije
Naivni Bajes	61.44
KNN	56.20
Stablo odlucivanja	53.12
Logisticka regresija	65.53
Voting	63.25
Bagging	65.45
Random forest	59.78

In [312…
```python
model_knn_new = KNeighborsClassifier(n_neighbors=100)
model_dt_new = DecisionTreeClassifier(max_depth=2)
model_voting = VotingClassifier(voting='soft', estimators=[('knn',model_knn_new),('tree', model_dt_new), ('log'
lista_new = list()
for model in [model_nb, model_knn_new, model_dt_new, model_log, model_voting, model_bagging, model_rf]:
    scores = cross_val_score(model, X_train_f, Y_train_f, cv=10, scoring='roc_auc')
    lista_new.append(round(np.mean(scores)*100, 2))
rez_optimizovano['Nakon optimizacije'] = lista_new
```

In [313…
```python
rez_optimizovano
```

Out[313]:

Algoritam	Pre optimizacije	Nakon optimizacije
Naivni Bajes	61.44	61.44
KNN	56.20	61.19
Stablo odlucivanja	53.12	63.08
Logisticka regresija	65.53	65.53
Voting	63.25	65.12
Bagging	65.45	65.38
Random forest	59.78	59.01

In [316…
```python
# Analizom svih rezultata dolazimo do zakljucka da nam Logisticka regresija daje najbolje rezultate i da cemo n
# kao model koji cemo korstiti za predvidjanje
```