DATA SCIENCE

-

PROJECT

AUTOSCOUT24

PROCEDURE:

- ➔ dataset Autoscout24
- ➔ descriptive analysis of the data set
- ➔ data cleaning
- ➔ correlations
- ➔ visualization Tableau
- ➔ plots Jupyter Notebook
- ➔ Supervised Learning (Regression)
- ➔ Supervised Learning with PCA

DATA SET:

PRELIMINARY CONSIDERATIONS:

➔ Autoscout24 data set on car sales and vehicle data from 2011 to 2021 - it contains basic information such as make, model, mileage, horsepower, etc. with the label “price”.

➔ AutoScout24 is the largest online car market in Europe. With AutoScout24, users can buy and sell used and new cars.
The used car market has developed in different directions in recent years. The reasons for this are diverse and cannot be reduced to Corona alone. It will be all the more important in the future to respond to this in a timely manner and with the right strategies.

AIM:

➔ This project is about analyzing and visualizing the data set. The cleaned data is trained using algorithms from the field of supervised learning (regression) in the form of a machine learning model so that precise price predictions can then be made.

FEATURES:

The data set contains 9 features, has 46405 samples and, with the label “price”, provides us with information about the price at which a car was sold:

mileage	mileage vehicle
make	brand
model	model
fuel	fuel type
gear	vehicle transmission
offerType	type of offer
price	selling price
hp	engine power
year	construction year

DESCRIPTIVE ANALYSIS OF THE DATASET:

➢ **Excerpt from the data set:**

	mileage	make	model	fuel	gear	offerType	price	hp	year
0	235000	BMW	316	Diesel	Manual	Used	6800	116.0	2011
1	92800	Volkswagen	Golf	Gasoline	Manual	Used	6877	122.0	2011
2	149300	SEAT	Exeo	Gasoline	Manual	Used	6900	160.0	2011
3	96200	Renault	Megane	Gasoline	Manual	Used	6950	110.0	2011
4	156000	Peugeot	308	Gasoline	Manual	Used	6950	156.0	2011

DATA CLEANING:

- **UNIQUE VALUES:**

mileage	20117
make	77
model	841
fuel	11
gear	3
offerType	5
price	6668
hp	328
year	11

- **NULL VALUES:**

mileage	0
make	0
model	143
fuel	0
gear	182
offerType	0
price	0
hp	29
year	0

➢ **The data set has null values in the columns„model“, “hp” and“gear”:**

```
Column "model" --> fill null values with "Different":

df01["model"] = df01["model"].fillna("Different")
```

```
Delete null values from column "hp":

df01.drop(df01[df01["hp"].isnull()].index, inplace=True)
```

```
Column "gear" --> fill null values with "Manual":

df01["gear"] = df01["gear"].fillna("Manual")
```

no more null values —>

```
mileage      0
make         0
model        0
fuel         0
gear         0
offerType    0
price        0
hp           0
year         0
```

➢ Checking and cleaning up outliers in the data set:

- **mileage:**

```
Checking mileage > 900.000:

df01[df01["mileage"] > 900000]
```

	mileage	make	model	fuel	gear	offerType	price	hp	year
16869	1111111	Opel	Karl	Gasoline	Manual	Demonstration	10490	73.0	2019
38049	999999	BMW	320	-/- (Fuel)	NaN	Used	1999	NaN	2014

```
Delete mileage > 900.000 - Opel Karl and BMW 320:

df01.drop(df01[df01["mileage"] > 999000].index, inplace=True)
```

- **price:**

```
Checking price > 800.000:

df01[df01["price"]  > 800000]

```

	mileage	make	model	fuel	gear	offerType	price	hp	year
21675	431	Ferrari	F12	Gasoline	Automatic	Used	1199900	775.0	2017

```
Delete Ferrari from "price":

df01.drop(df01[df01["price"] >1000000].index, inplace=True)
```

- **delete “make” - samples “Trailer-Anhänger”:**

```
Delete "Trailer-Anhänger" from make:

df01.drop(df01[df01["make"] == "Trailer-Anhänger"].index, inplace=True)
```

- **add new column “carAge”:**

```
Add column "carAge":

df01["carAge"] = 2021 - df01["year"]
```

- **delete column “year”:**

```
Delete column "year":

df01.drop("year", axis=1, inplace=True)
```

➢ Categorical values:

- <u>transform the one-dimensional arrays of the categorical features into lists:</u>

```
MAKE
liste_make = []

arr01 = np.array(df01["make"])

for i in arr01:
    liste_make.append(i)
```

```
MODEL
liste_model = []

arr02 = np.array(df01["model"])

for i in arr02:
    liste_model.append(i)
```

```
GEAR
liste_gear = []

arr04 = np.array(df01["gear"])

for i in arr04:
        liste_gear.append(i)
```

```
FUEL
liste_fuel = []

arr03 = np.array(df01["fuel"])

for i in arr03:
        liste_fuel.append(i)
```

```
OFFERTYPE
liste_offerType = []

arr05 = np.array(df01["offerType"])

for i in arr05:
        liste_offerType.append(i)
```

- **converting features - import, initialize and fit/transform in one step:**

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()


encoded_make = le.fit_transform(liste_make)

encoded_model = le.fit_transform(liste_model)

encoded_fuel = le.fit_transform(liste_fuel)

encoded_gear = le.fit_transform(liste_gear)

encoded_offerType = le.fit_transform(liste_offerType)
```

- **filling the new features with numerical values:**

```
df01["encoded_make"] = encoded_make
df01["encoded_model"] = encoded_model
df01["encoded_fuel"] = encoded_fuel
df01["encoded_gear"] = encoded_gear
df01["encoded_offerType"] = encoded_offerType
```

- **transformed numeric features:**

mileage	make	model	fuel	gear	offerType	price	hp	carAge	encoded_make	encoded_model	encoded_fuel	encoded_gear	encoded_offerType
235000	BMW	316	Diesel	Manual	Used	6800	116.0	10	8	33	2	1	4
92800	Volkswagen	Golf	Gasoline	Manual	Used	6877	122.0	10	72	396	7	1	4
149300	SEAT	Exeo	Gasoline	Manual	Used	6900	160.0	10	63	324	7	1	4
96200	Renault	Megane	Gasoline	Manual	Used	6950	110.0	10	61	508	7	1	4
156000	Peugeot	308	Gasoline	Manual	Used	6950	156.0	10	56	32	7	1	4

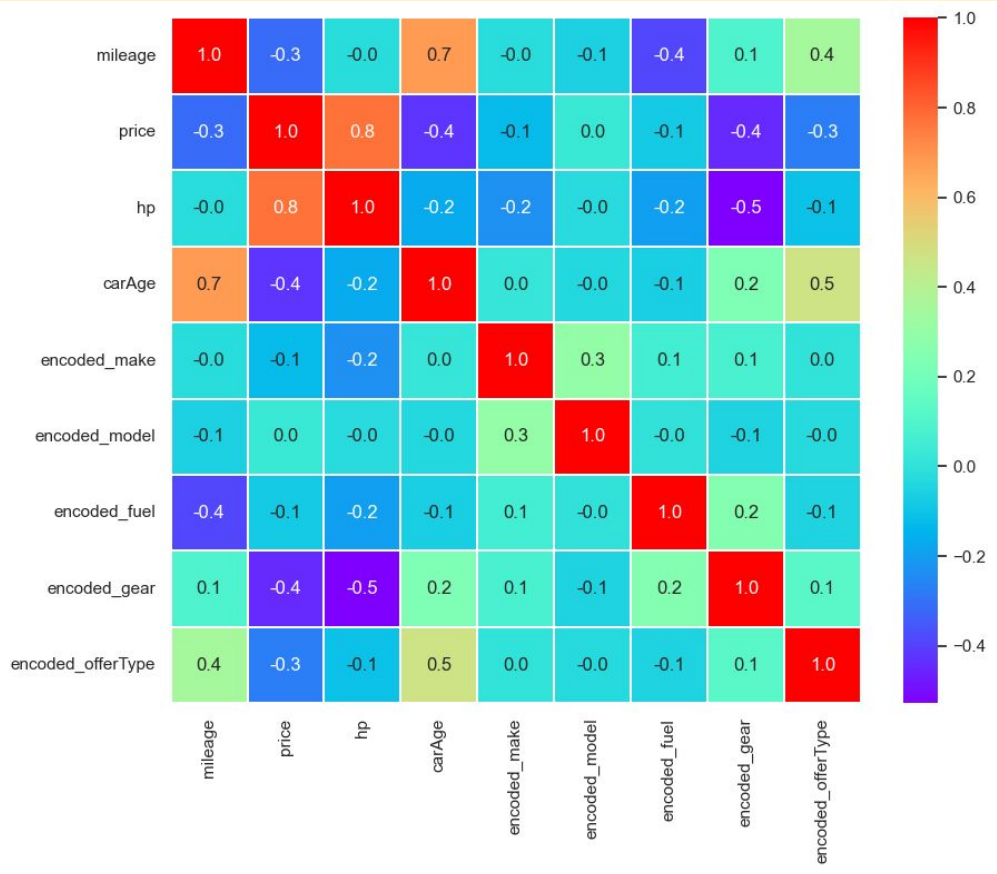
➢ DESCRIPTIVE VALUES:

	mileage	price	hp	carAge	encoded_make	encoded_model	encoded_fuel	encoded_gear	encoded_offerType
count	46370.000000	46370.000000	46370.000000	46370.000000	46370.000000	46370.000000	46370.000000	46370.000000	46370.000000
mean	71152.333146	16549.427367	132.989648	4.987492	47.120832	420.340802	5.228639	0.660988	3.663468
std	62268.411408	18510.702873	75.385055	3.154988	21.620313	255.886540	2.363129	0.475924	0.989708
min	0.000000	1100.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	19837.750000	7490.000000	86.000000	2.000000	29.000000	184.000000	2.000000	0.000000	4.000000
50%	60000.000000	10999.000000	116.000000	5.000000	54.000000	401.000000	7.000000	1.000000	4.000000
75%	105000.000000	19490.000000	150.000000	8.000000	64.000000	637.000000	7.000000	1.000000	4.000000
max	699000.000000	717078.000000	850.000000	10.000000	75.000000	839.000000	10.000000	2.000000	4.000000

CORRELATIONS:

	mileage	price	hp	carAge	encoded_make	encoded_model	encoded_fuel	encoded_gear	encoded_offerType
mileage	1.000000	-0.315705	-0.014821	0.679991	-0.018055	-0.061605	-0.385329	0.088951	0.354155
price	-0.315705	1.000000	0.768742	-0.422631	-0.125835	0.032392	-0.084240	-0.448837	-0.276315
hp	-0.014821	0.768742	1.000000	-0.167375	-0.230519	-0.022735	-0.193923	-0.528100	-0.107548
carAge	0.679991	-0.422631	-0.167375	1.000000	0.016232	-0.036476	-0.067132	0.235296	0.465645
encoded_make	-0.018055	-0.125835	-0.230519	0.016232	1.000000	0.300344	0.062236	0.071128	0.007280
encoded_model	-0.061605	0.032392	-0.022735	-0.036476	0.300344	1.000000	-0.002338	-0.054958	-0.028605
encoded_fuel	-0.385329	-0.084240	-0.193923	-0.067132	0.062236	-0.002338	1.000000	0.248442	-0.055401
encoded_gear	0.088951	-0.448837	-0.528100	0.235296	0.071128	-0.054958	0.248442	1.000000	0.124615
encoded_offerType	0.354155	-0.276315	-0.107548	0.465645	0.007280	-0.028605	-0.055401	0.124615	1.000000

- **heatmap correlations:**



- **THE MOST IMPORTANT CORRELATIONS:**

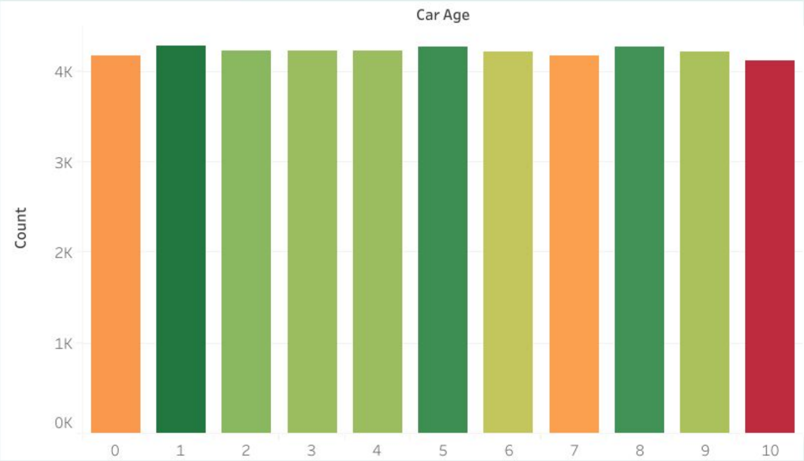
CORRELATION 0.8:	price - hp
CORRELATION 0.7:	carAge - mileage
CORRELATION 0.5:	carAge - offerType
CORRELATION 0.4:	encoded_offerType - mileage
CORRELATION 0.3:	encoded_mark - encoded_model
CORRELATION -0.5:	encoded_gear - hp
CORRELATION -0.4:	encoded_fuel - mileage
	encoded_gear - price
	price - carAge
CORRELATION -0.3:	encoded_offerType - price
	mileage - price

VISUALIZATIONS TABLEAU:

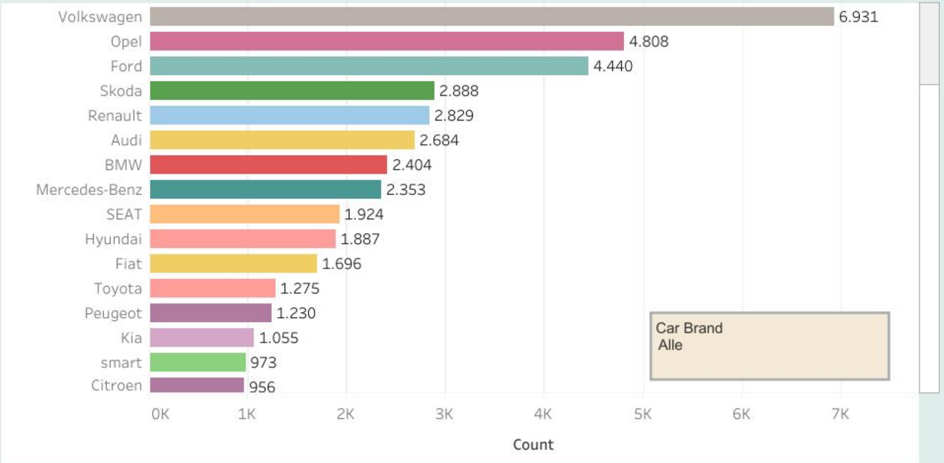
LINK TABLEAU: https://public.tableau.com/app/profile/manuela.holzner/viz/Autoscout24/04Sales

GENERAL INFORMATIONS

Sold Cars - Car Age

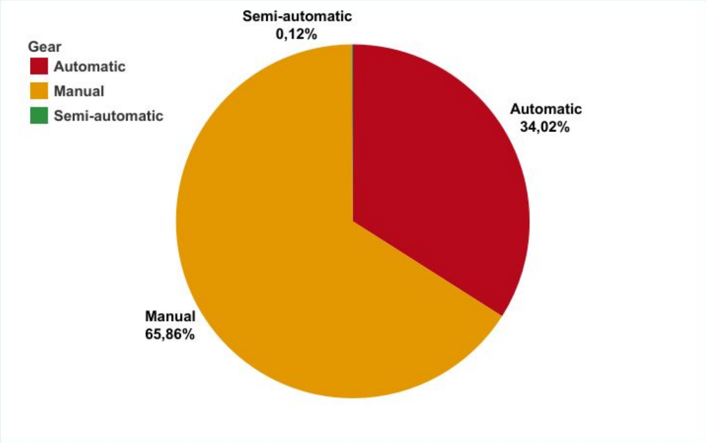


Sold Cars - Car Brand

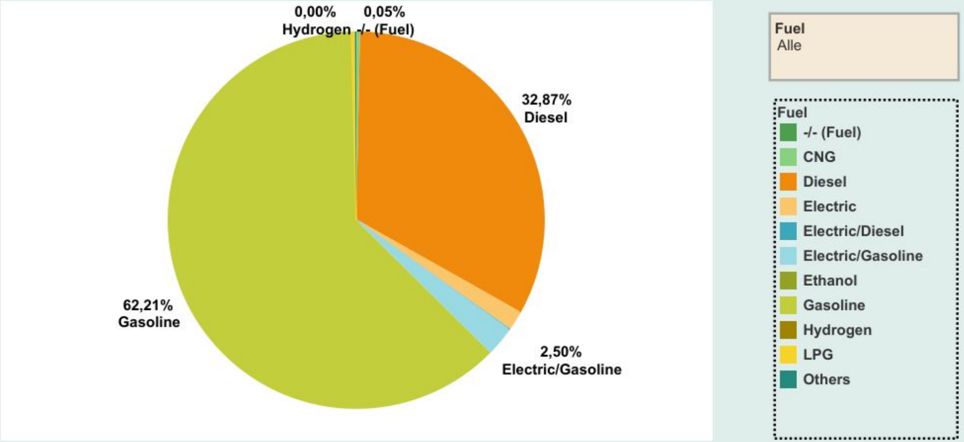


GENERAL INFORMATIONS

Sold Cars - Gear Type

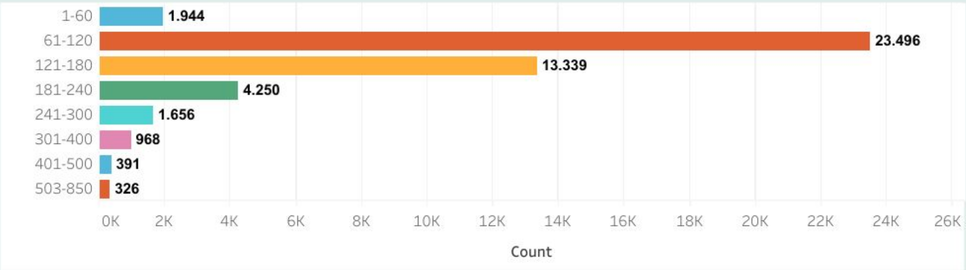


Sold Cars - Fuel Type

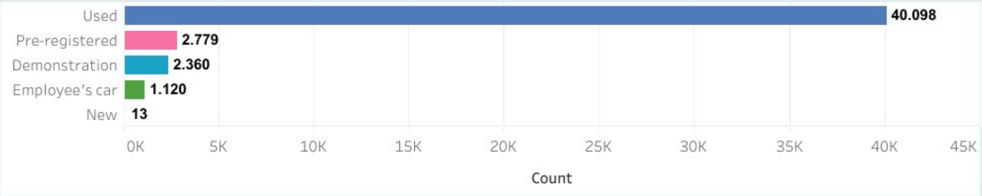


GENERAL INFORMATIONS

Sold Cars - Horsepower



Sold Cars - Offer Type



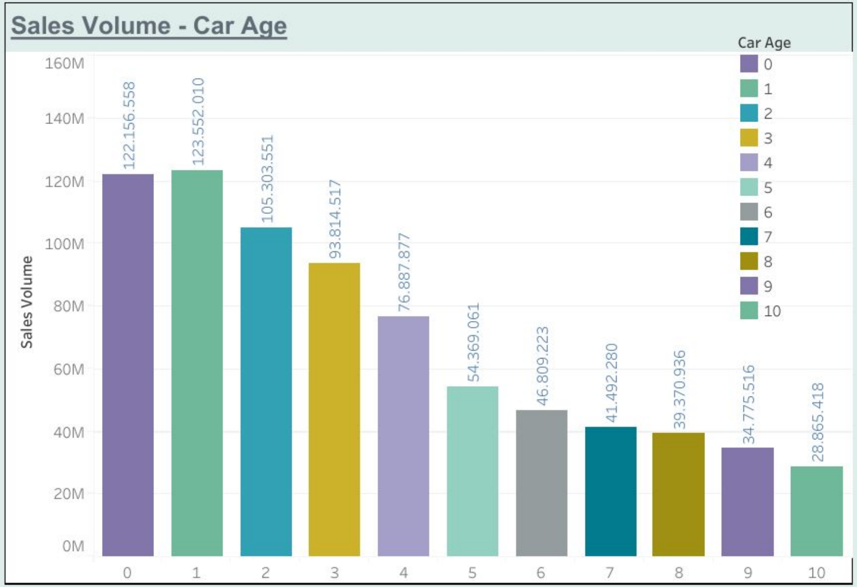
Overview Car Brand - Sold Model

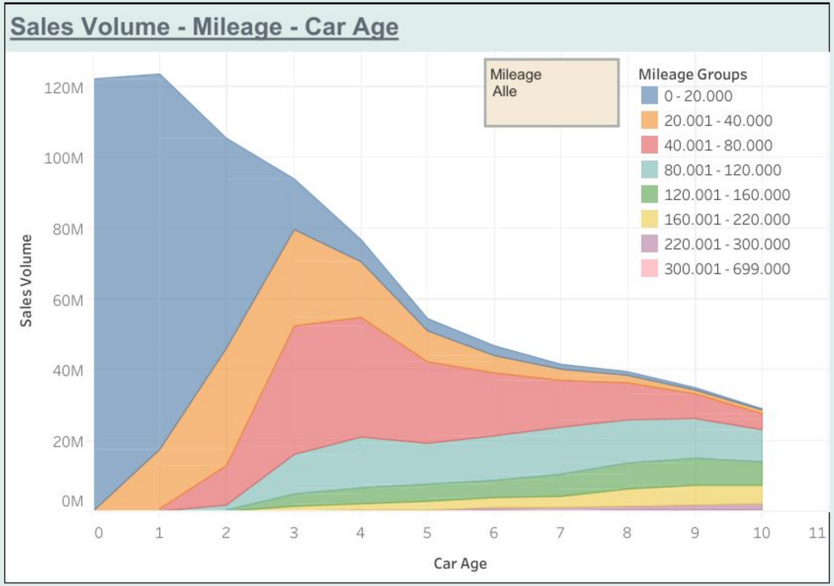
Make	Model	
9ff	Different	1
Abarth	500	12
	595	11
	595 Competizione	4
	595 Turismo	5
	595C	7
	695	2
	Grande Punto	1
	Punto EVO	1
Aixam	City	2
Alfa	Romeo 4C	1
	Romeo 159	5
	Romeo Giulia	20
	Romeo Giulietta	50
	Romeo MiTo	27
	Romeo Quadrifoglio	1
	Romeo Sportwagon	1
	Romeo Stelvio	27
Alpina	B3	5
	B5	1
	B7	1

Make
Alle

Model
Alle

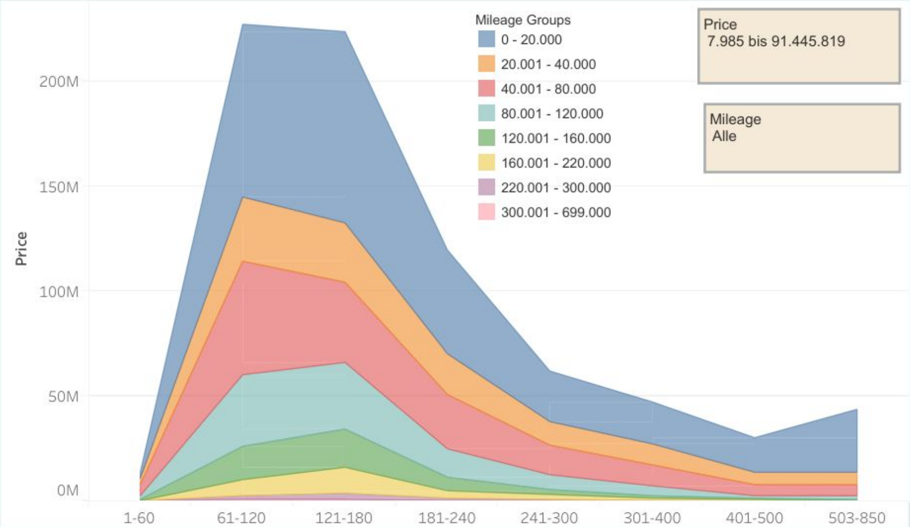
SALES VOLUMES



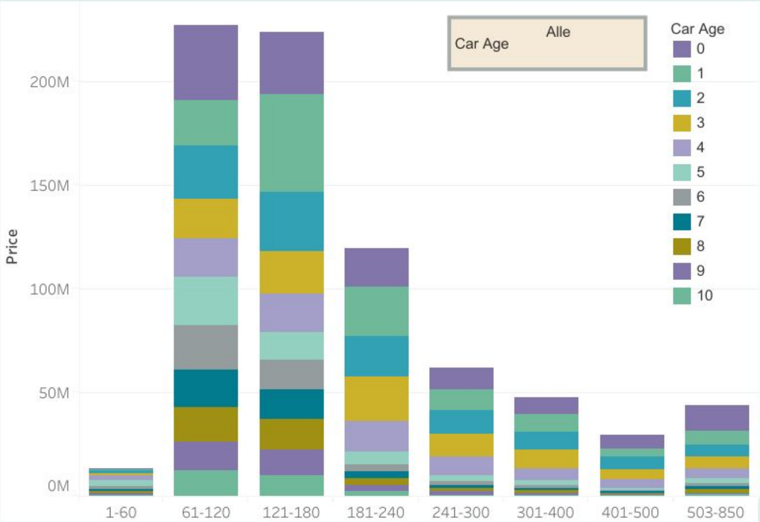


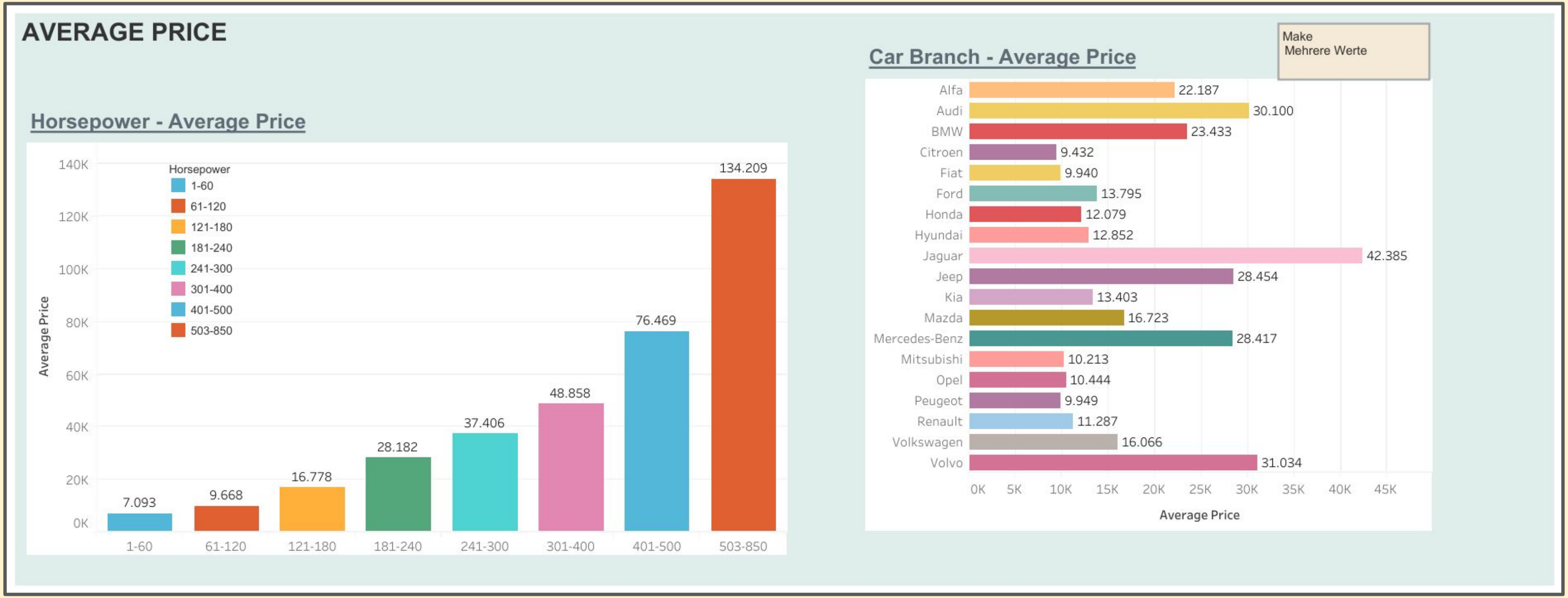
HORSEPOWER

Horsepower - Price - Mileage



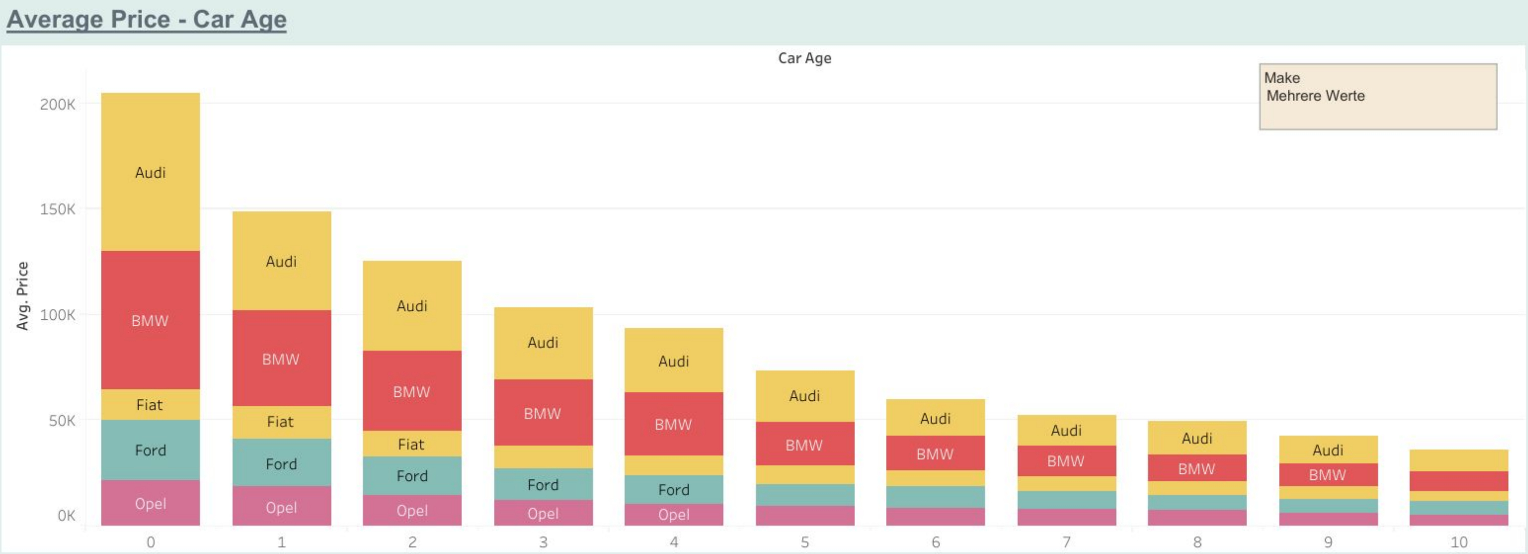
Horsepower - Price - Car Age





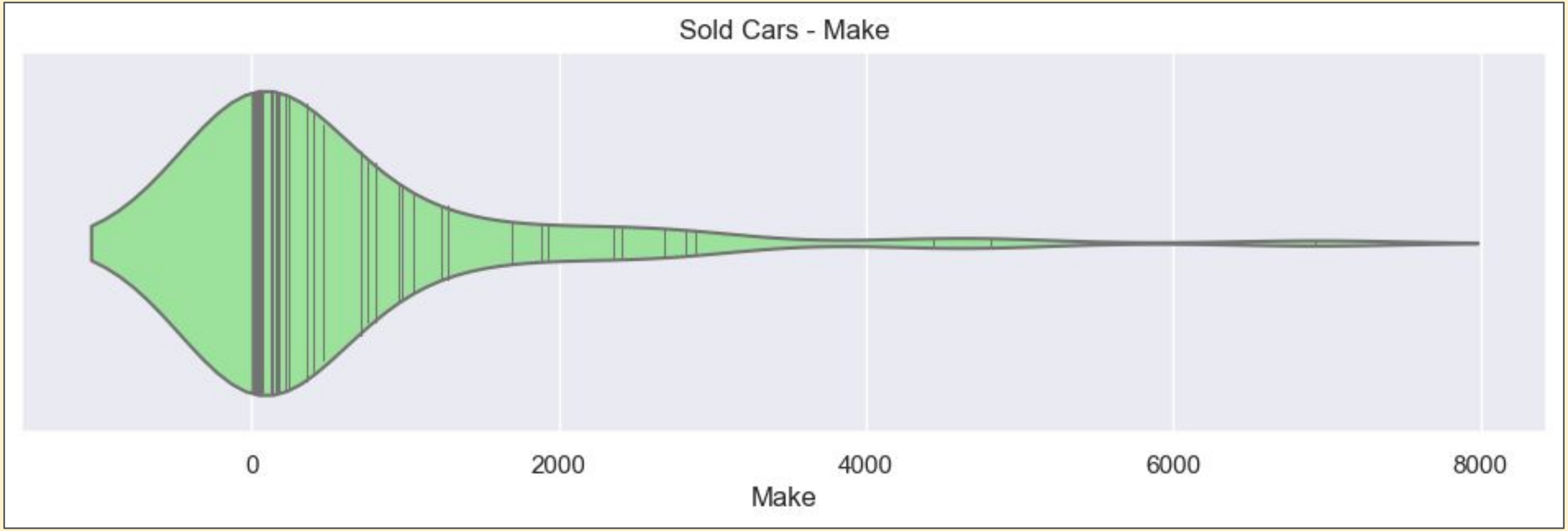
AVERAGE PRICE
Horsepower - Average Price
Horsepower
1-60
61-120
121-180
181-240
241-300
301-400
401-500
503-850
Average Price
140K
120K
100K
80K
60K
40K
20K
0K
7.093
9.668
16.778
28.182
37.406
48.858
76.469
134.209
Car Branch - Average Price
Make
Mehrere Werte
Alfa 22.187
Audi 30.100
BMW 23.433
Citroen 9.432
Fiat 9.940
Ford 13.795
Honda 12.079
Hyundai 12.852
Jaguar 42.385
Jeep 28.454
Kia 13.403
Mazda 16.723
Mercedes-Benz 28.417
Mitsubishi 10.213
Opel 10.444
Peugeot 9.949
Renault 11.287
Volkswagen 16.066
Volvo 31.034
0K 5K 10K 15K 20K 25K 30K 35K 40K 45K
Average Price

AVERAGE PRICE

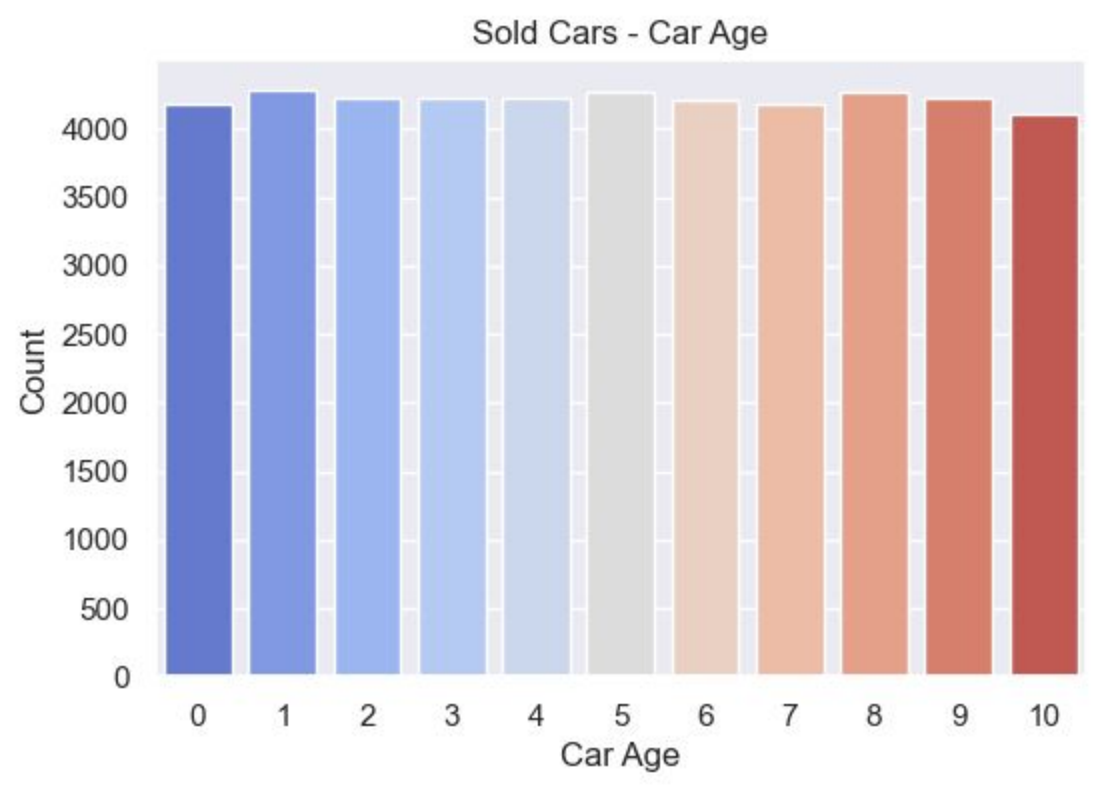


PLOTS JUPYTER NOTEBOOK:

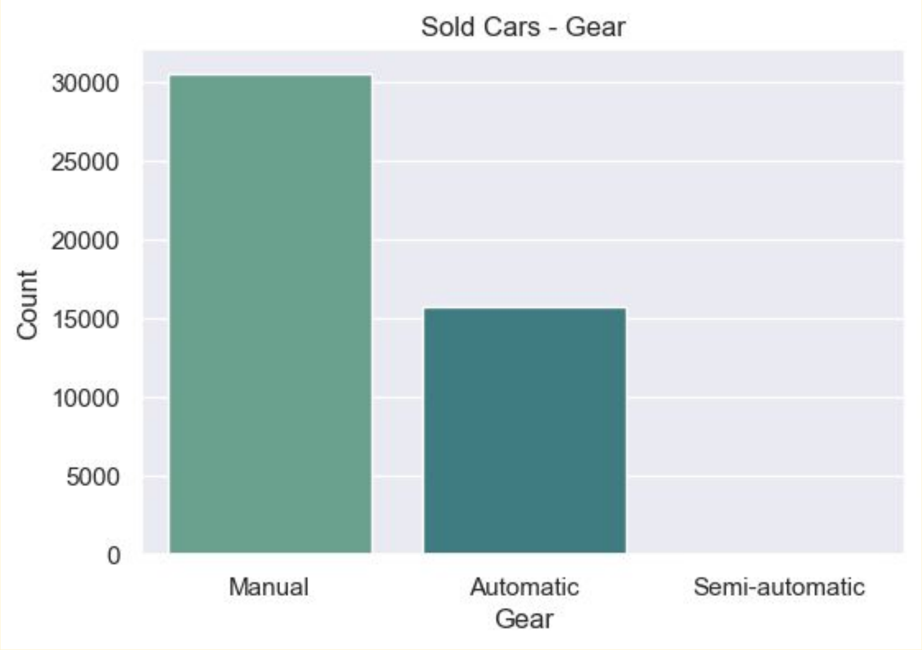
- **distribution of sold car brands:**



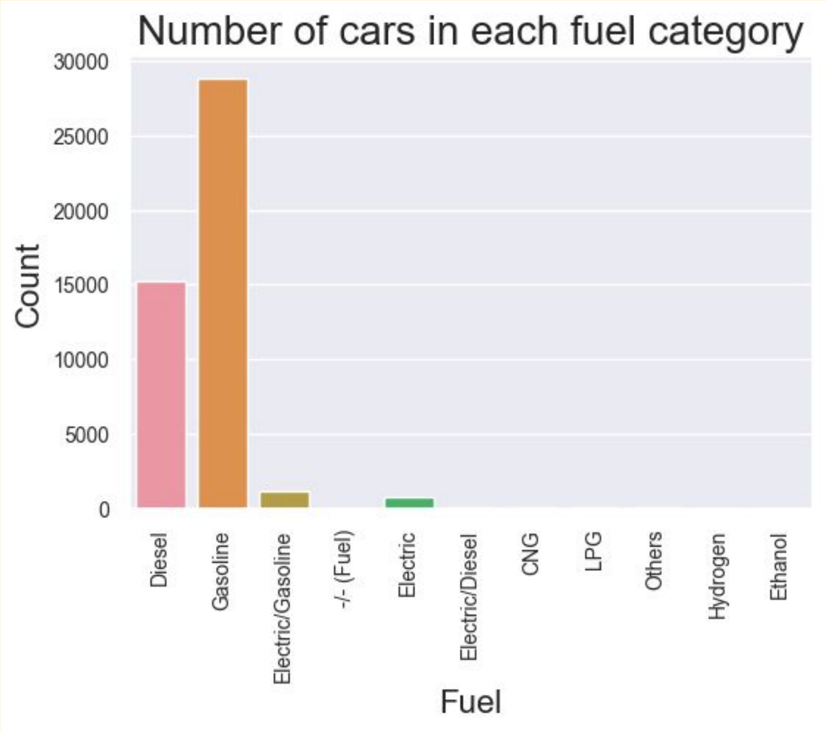
- **<u>distribution of vehicles sold - age of vehicles:</u>**

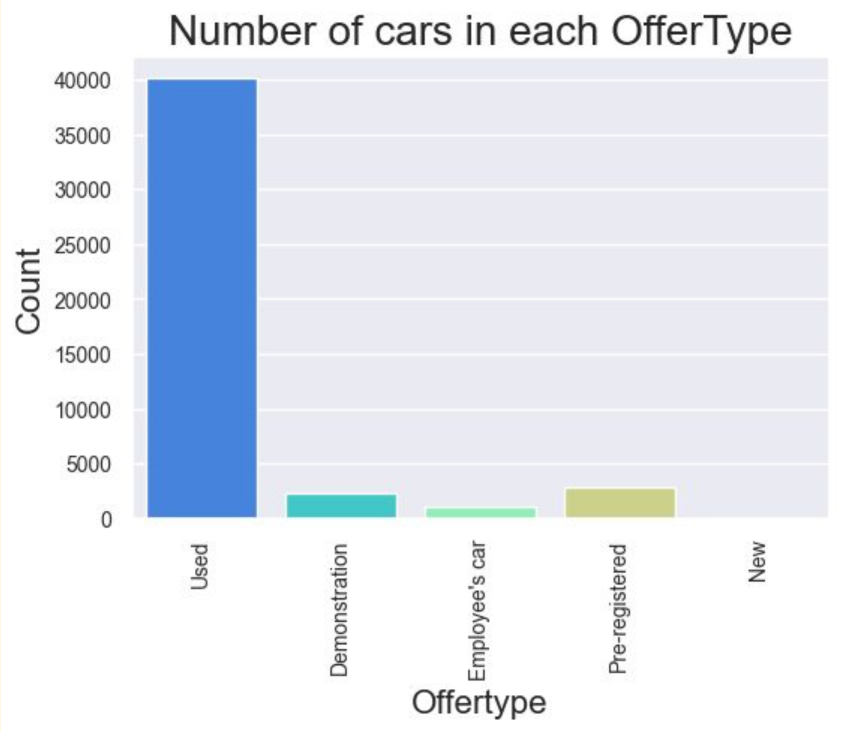


- **distribution of vehicles sold - price/age and gear:**



- **distribution of vehicles sold - fuel type and type of offer:**





SUPERVISED LEARNING - REGRESSION:

- **the 5 best-selling brands should be used for processing in machine learning:**

```
df01["make"].value_counts().head(5)
```

```
make
Volkswagen    6931
Opel          4808
Ford          4440
Skoda         2888
Renault       2829
Name: count, dtype: int64
```

```
df01 = df01[(df01["make"]  == "Volkswagen") | (df01["make"]  == "Opel") | (df01["make"]  == "Ford")
            | (df01["make"]  == "Skoda") | (df01["make"]  == "Renault")]
```

```
df01.sample(5)
```

mileage	make	model	fuel	gear	offerType	price	hp	carAge	encoded_make	encoded_model	encoded_fuel	encoded_gear	encoded_offerTy
43000	Opel	Mokka	Gasoline	Manual	Used	11950	140.0	7	54	517	7	1	
94250	Volkswagen	Phaeton	Gasoline	Automatic	Used	34822	334.0	5	72	562	7	0	
228418	Volkswagen	Golf	Diesel	Manual	Used	3899	105.0	10	72	396	2	1	
158000	Ford	Focus	Diesel	Manual	Used	7590	120.0	4	29	338	2	1	
104000	Opel	Corsa	Gasoline	Manual	Used	4690	87.0	10	54	274	7	1	

- <u>**AVERAGE PRICE BY BRAND:**</u>

```python
vw = df01[(df01["make"]  == "Volkswagen")]
opel = df01[(df01["make"]  == "Opel")]
ford = df01[(df01["make"]  == "Ford")]
skoda = df01[(df01["make"]  == "Skoda")]
renault = df01[(df01["make"]  == "Renault")]
```

```python
print("Average Price Volkswagen:",vw["price"].mean().round(0))
print("Average Price Opel:      ",opel["price"].mean().round(0))
print("Average Price Ford:      ",ford["price"].mean().round(0))
print("Average Price Skoda:     ",skoda["price"].mean().round(0))
print("Average Price Renault:   ",renault["price"].mean().round(0))
```

```
Average Price Volkswagen: 16066.0
Average Price Opel:       10444.0
Average Price Ford:       13795.0
Average Price Skoda:      13726.0
Average Price Renault:    11287.0
```

➢ Training and Predicting:

- <u>**Algorithms - Variables - Train Test Split:**</u>

```
NUMERIC PREDICTION ALGORITHMS:

Linear Regression
Decision Tree
Randorm Forest
```

```
8 FEATURES - 21.896 SAMPLES

X01.shape

```

```
(21896, 8)
```

```
VARIABLES FOR TRAINING UND PREDICTION

X01 = df01.drop(["make", "model", "fuel", "gear", "offerType", "price"], axis=1)
y01 = df01["price"]
```

```
TRAIN TEST SPLIT

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X01, y01, test_size = 0.20, random_state = 101)
```

- **Import Algorithms, Train and Predict:**

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor

lin = LinearRegression()
dec = DecisionTreeRegressor()
rfc = RandomForestRegressor()
```

```
TRAIN

lin01 = lin.fit(X_train, y_train)
dec01 = DecisionTreeRegressor(random_state = 101).fit(X_train, y_train)
rfc01 = RandomForestRegressor(random_state = 101, n_estimators = 1000).fit(X_train, y_train)
```

```
PREDICT

pred_lin01 = lin01.predict(X_test)
pred_dec01 = dec01.predict(X_test)
pred_rfc01 = rfc01.predict(X_test)
```

- **check samples - predictions from the data set:**

mileage	make	model	fuel	gear	offerType	price	hp	carAge	encoded_make	encoded_model	encoded_fuel	encoded_gear	encoded_o
70000	Skoda	Roomster	Gasoline	Automatic	Used	9890	105.0	8	64	629	7	0	
10000	Volkswagen	Golf	Gasoline	Manual	Demonstration	28890	150.0	1	72	396	7	1	
20	Renault	Kangoo	Gasoline	Automatic	Used	15070	114.0	2	61	452	7	0	
15750	Volkswagen	Caddy	Diesel	Automatic	Demonstration	28750	150.0	1	72	219	2	0	
16551	Volkswagen	Polo	Gasoline	Manual	Used	11944	65.0	3	72	566	7	1	

```python
print("Real Price Skoda Roomster: 9890\n")
print("Prediction Linear Regression:  ",lin01.predict([[70000, 105, 8, 64, 629, 7, 0, 4]]).round(0))
print("Prediction Decision Tree:      ", dec01.predict([[70000, 105, 8, 64, 629, 7, 0, 4]]).round(0))
print("Prediction Random Forest:      ", rfc01.predict([[70000, 105, 8, 64, 629, 7, 0, 4]]).round(0))
```

```
Real Price Skoda Roomster: 9890

Prediction Linear Regression:   [10608.]
Prediction Decision Tree:       [9890.]
Prediction Random Forest:       [9684.]
```

```python
print("Real Price Volkswagen Golf: 28890\n")
print("Prediction Linear Regression:  ",lin01.predict([[10000, 150.0, 1, 72, 396, 7, 1, 0]]).round(0))
print("Prediction Decision Tree:      ", dec01.predict([[10000, 150.0, 1, 72, 396, 7, 1, 0]]).round(0))
print("Prediction Random Forest:      ", rfc01.predict([[10000, 150.0, 1, 72, 396, 7, 1, 0]]).round(0))
```

```
Real Price Volkswagen Golf: 28890

Prediction Linear Regression:   [27177.]
Prediction Decision Tree:       [28890.]
Prediction Random Forest:       [28747.]
```

```python
print("Real Price Renault Kango: 15070\n")
print("Prediction Linear Regression:  ",lin01.predict([[20, 114.0, 2, 61, 452, 7, 0, 4]]).round(0))
print("Prediction Decision Tree:      ", dec01.predict([[20, 114.0, 2, 61, 452, 7, 0, 4]]).round(0))
print("Prediction Random Forest:      ", rfc01.predict([[20, 114.0, 2, 61, 452, 7, 0, 4]]).round(0))
```

```
Real Price Renault Kango: 15070

Prediction Linear Regression:   [19622.]
Prediction Decision Tree:       [14917.]
Prediction Random Forest:       [15062.]
```

```
print("Real Price Volkswagen Caddy: 28750\n")
print("Prediction Linear Regression:  ",lin01.predict([[15750, 150.0, 1, 72, 219, 2, 0, 0]]).round(0))
print("Prediction Decision Tree:      ", dec01.predict([[15750, 150.0, 1, 72, 219, 2, 0, 0]]).round(0))
print("Prediction Random Forest:      ", rfc01.predict([[15750, 150.0, 1, 72, 219, 2, 0, 0]]).round(0))
```

```
Real Price Volkswagen Caddy: 28750

Prediction Linear Regression:   [30356.]
Prediction Decision Tree:       [28950.]
Prediction Random Forest:       [29576.]
```

```
print("Real Price Volkswagen Polo: 11944\n")
print("Prediction Linear Regression:  ",lin01.predict([[16551, 65.0, 3, 72, 566, 7, 1, 4]]).round(0))
print("Prediction Decision Tree:      ", dec01.predict([[16551, 65.0, 3, 72, 566, 7, 1, 4]]).round(0))
print("Prediction Random Forest:      ", rfc01.predict([[16551, 65.0, 3, 72, 566, 7, 1, 4]]).round(0))
```

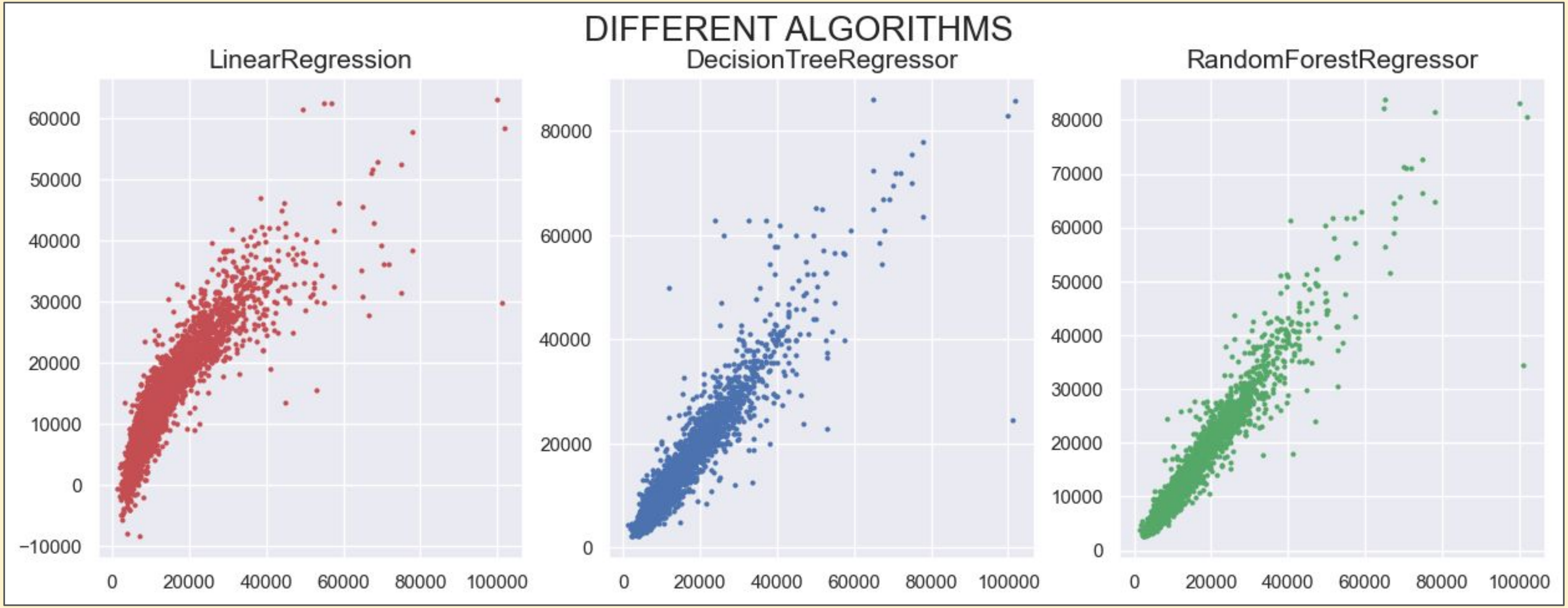
```
Real Price Volkswagen Polo: 11944

Prediction Linear Regression:   [11484.]
Prediction Decision Tree:       [11944.]
Prediction Random Forest:       [11469.]
```

➢ Validation:

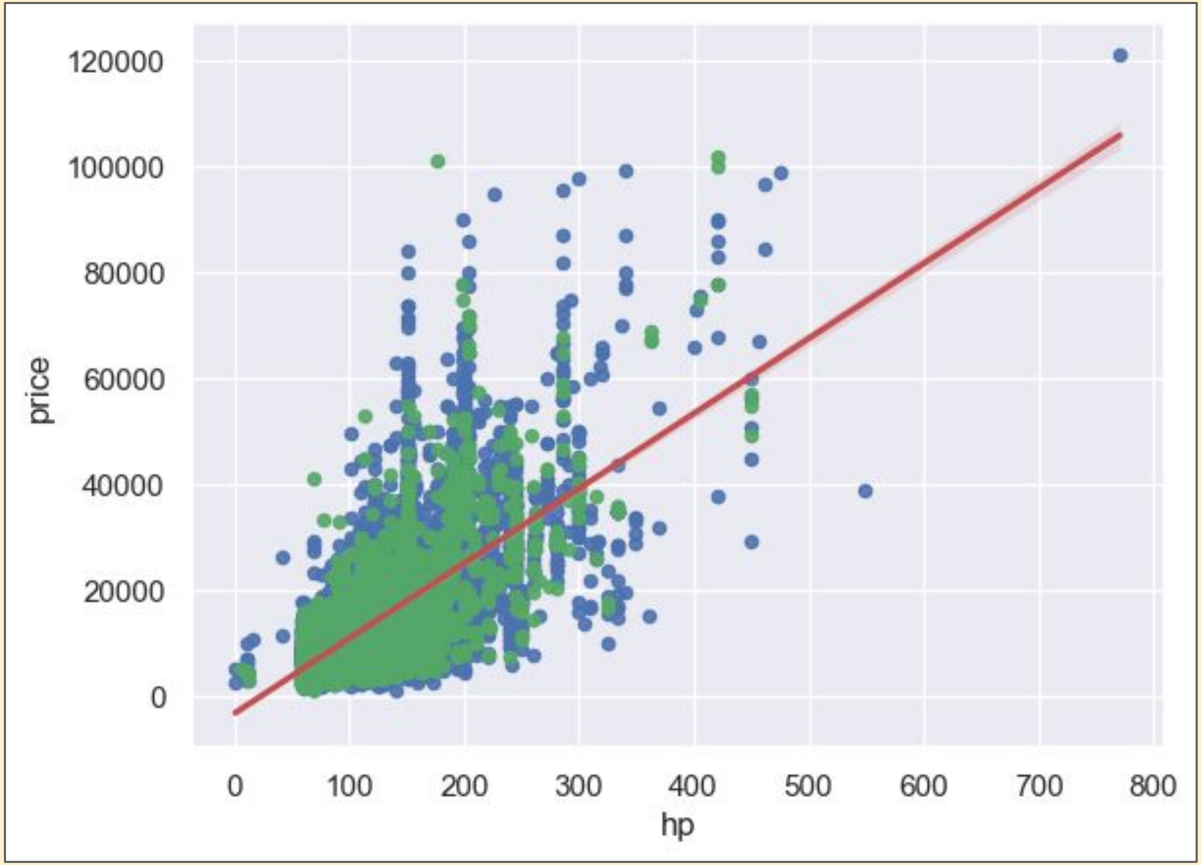
- **metrics:**

```
LINEAR REGRESSION:
Mean absolute error: 2717.49
Mean squared error : 18209427.83
Root squared error : 4267.25

DECISION TREE:
Mean absolute error: 1678.54
Mean squared error : 10189065.05
Root squared error : 3192.03

RANDOM FOREST:
Mean absolute error: 1328.13
Mean squared error : 6113752.53
Root squared error : 2472.6
```

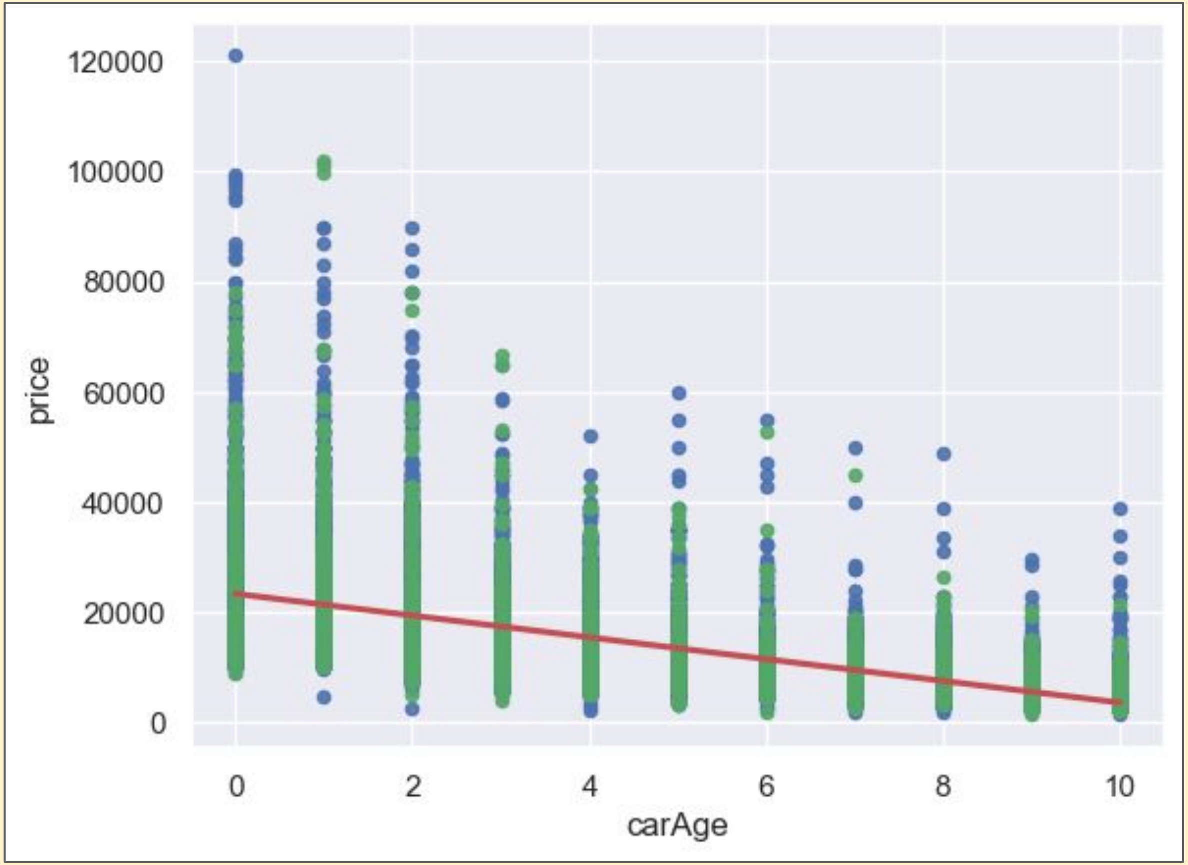
- **visualization algorithms:**



- **comparison - feature with a strange correlation "hp" with label "price":**

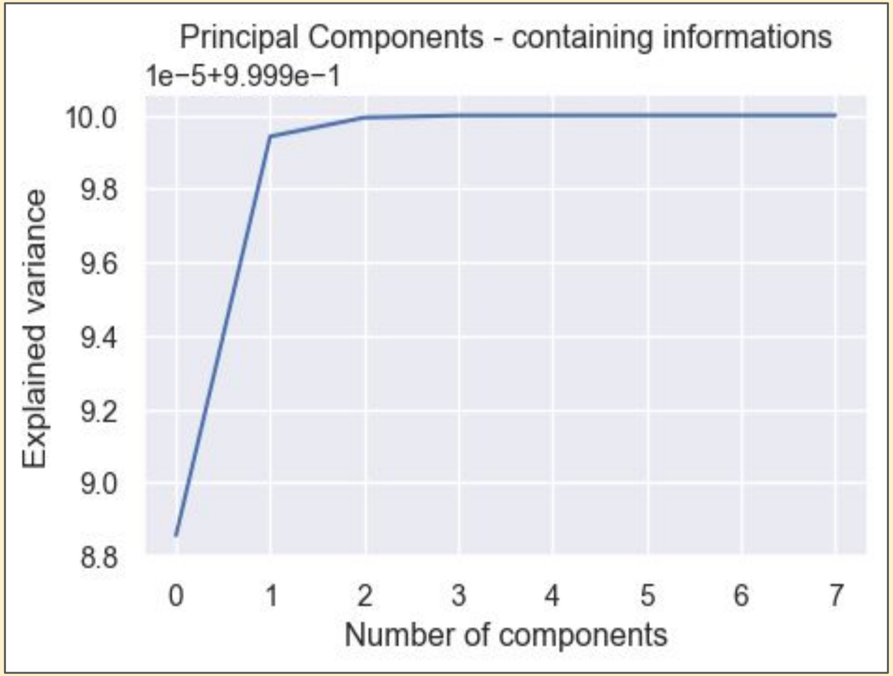


- **comparison - feature with a strange correlation "carAge" with label "price":**



SUPERVISED LEARNING - PCA:

- **Check Principal Components:**




```
First component contains over 100 % of the information

for i,value in enumerate(pca_check.explained_variance_ratio_):
    print(f"{i+1}. Principal Component explains
          {value*100:.4f}% of the variance")
```

```
1. Principal Component explains 99.9989% of the variance
2. Principal Component explains 0.0011% of the variance
3. Principal Component explains 0.0001% of the variance
4. Principal Component explains 0.0000% of the variance
```

- **Standardscaler for variance:**

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
scaled_X01 = scaler.fit_transform(X01)
```

- **PCA with 1 component:**

```
pca = PCA(n_components=1, random_state=33)
```

- **PCA training and transforming:**

```
x_pca = pca.fit_transform(scaled_X01)
```

```
x_pca.shape
```

```
(21896, 1)
```

➢ Training and Predicting PCA:

- **Train Test Split - PCA 1 feature:**

```
NUMERIC PREDICTION ALGORITHMS:

Linear Regression
Decision Tree
Randorm Forest
```

```
1 FEATURE - 21.896 SAMPLES

X02.shape

```

```
(21896, 1)
```

```
X02 = x_pca
y02 = df01["price"]
```

```
TRAIN TEST SPLIT

X_train02, X_test02, y_train02, y_test02 = train_test_split(X02, y02, test_size = 0.20, random_state = 101)
```

- **Train and Predict:**

```
TRAIN

lin02 = lin.fit(X_train02, y_train02)
dec02 = DecisionTreeRegressor(random_state = 101).fit(X_train02, y_train02)
rfc02 = RandomForestRegressor(random_state = 101, n_estimators = 1000).fit(X_train02, y_train02)
```

```
PREDICT

pred_lin02 = lin02.predict(X_test02)
pred_dec02 = dec02.predict(X_test02)
pred_rfc02 = rfc02.predict(X_test02)
```

➢ **Validation:**

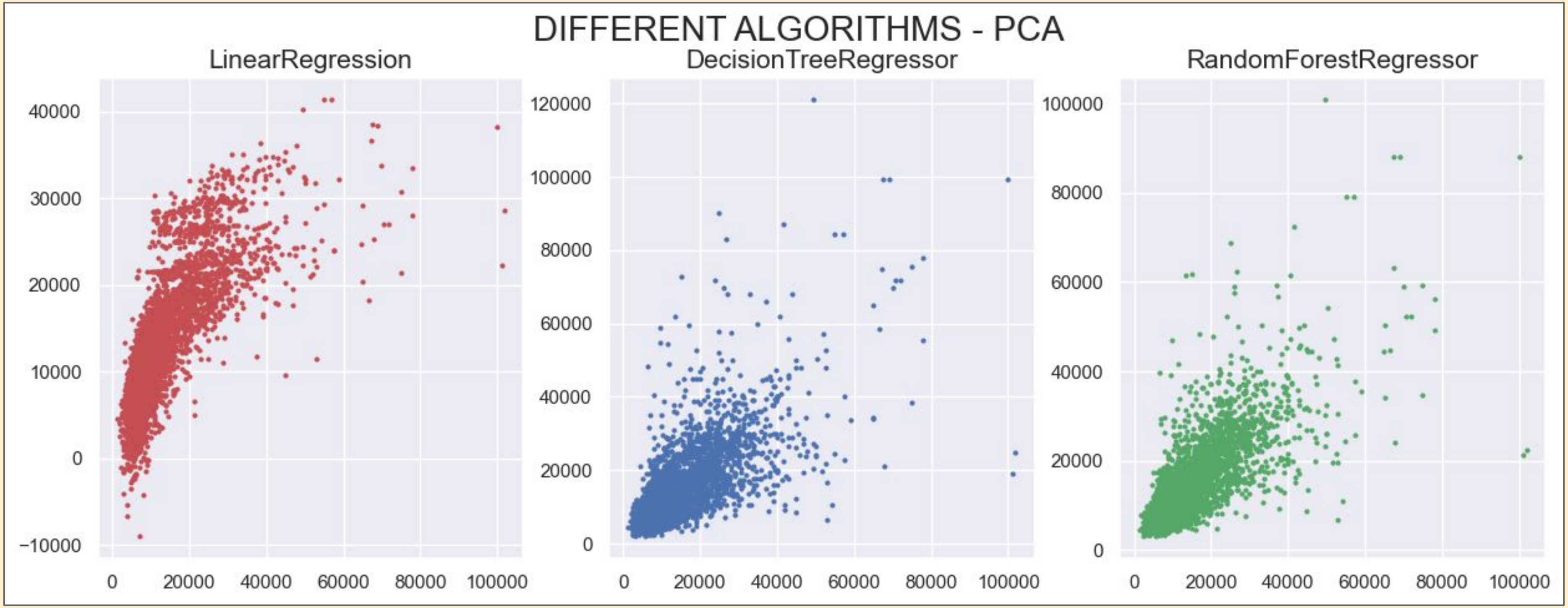
- **metrics PCA:**

```
LINEAR REGRESSION:
Mean absolute error: 3994.61
Mean squared error : 38287795.64
Root squared error : 6187.71

DECISION TREE:
Mean absolute error: 4188.57
Mean squared error : 51074508.75
Root squared error : 7146.64

RANDOM FOREST:
Mean absolute error: 3749.91
Mean squared error : 38313257.57
Root squared error : 6189.77
```

- **comparison metrics without PCA:**

```
LINEAR REGRESSION:
Mean absolute error: 2717.49
Mean squared error : 18209427.83
Root squared error : 4267.25

DECISION TREE:
Mean absolute error: 1678.54
Mean squared error : 10189065.05
Root squared error : 3192.03

RANDOM FOREST:
Mean absolute error: 1328.13
Mean squared error : 6113752.53
Root squared error : 2472.6
```

- **visualization algorithms PCA:**



CONCLUSION:

- ➔ the algorithms without PCA get good - very good results:
  - ❑ Decision Tree get very good results in the lower price segment
  - ❑ Random Forest get good - very good results in the lower price segment , in the higher price segment it is the BEST
  - ❑ Linear regression performs worst in prediction
- ➔ the algorithms with PCA still get satisfactory results, but significantly worse than those without PCA

```
LINEAR REGRESSION:
Mean absolute error: 2717.49
Mean squared error : 18209427.83
Root squared error : 4267.25

DECISION TREE:
Mean absolute error: 1678.54
Mean squared error : 10189065.05
Root squared error : 3192.03

RANDOM FOREST:
Mean absolute error: 1328.13
Mean squared error : 6113752.53
Root squared error : 2472.6
```

```
LINEAR REGRESSION:
Mean absolute error: 3994.61
Mean squared error : 38287795.64
Root squared error : 6187.71

DECISION TREE:
Mean absolute error: 4188.57
Mean squared error : 51074508.75
Root squared error : 7146.64

RANDOM FOREST:
Mean absolute error: 3749.91
Mean squared error : 38313257.57
Root squared error : 6189.77
```

OVERALL CONCLUSION:

➔ A manageable number of features allowed the data set to be processed and analyzed well:
- Null values could be easily equalized; only a few samples had to be deleted
- The “price” label was already there and therefore provided the direct target

➔ Very good results could be achieved in supervised learning with the features because the categorical values could be cleanly converted into numerical ones in order to work with the regression algorithms

➔ Evaluation of the 3 algorithms used:

Best Results: **RandomForestRegressor**

Minimally worse results: **DecisionTreeRegressor**

Worst results: **Linear Regression and Algorithms with PCA**