Prediction of used car value

Contents

Objective

The number of vehicles owned per person is increasing → **Activation of used car market**

To predict used car's value

	dateCrawled	name	seller	offerType
0	2016-03-24 11:52:17	Golf_3_1.6	privat	Angebot
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	privat	Angebot
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	privat	Angebot
3	2016-03-17 16:54:04	GOLF_4_1_4__3TÜRER	privat	Angebot
4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	privat	Angebot

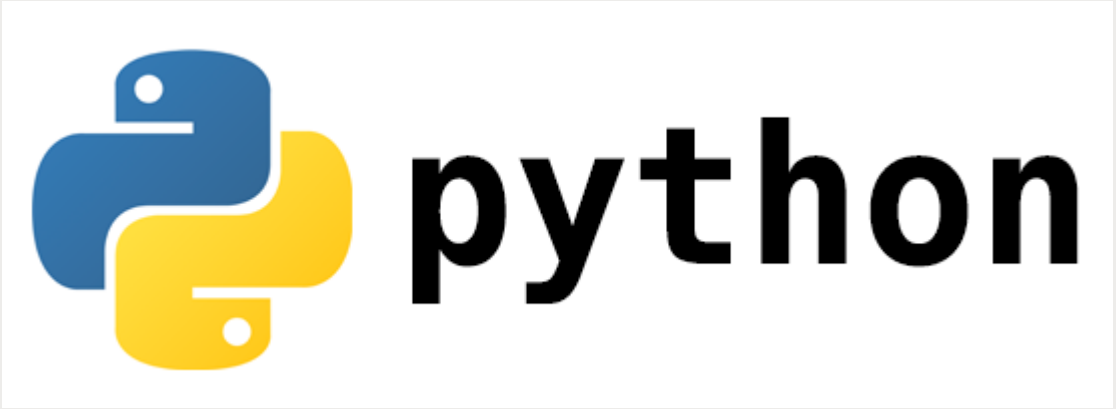
	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model
0	480	test	NaN	1993	manuell	0	golf
1	18300	test	coupe	2011	manuell	190	NaN
2	9800	test	suv	2004	automatik	163	grand
3	1500	test	kleinwagen	2001	manuell	75	golf
4	3600	test	kleinwagen	2008	manuell	69	fabia

	kilometer	monthOfRegistration	fuelType	brand	notRepairedDamage
0	150000	0	benzin	volkswagen	NaN
1	125000	5	diesel	audi	ja
2	125000	8	diesel	jeep	NaN
3	150000	6	benzin	volkswagen	nein
4	90000	7	diesel	skoda	nein

	dateCreated	nrOfPictures	postalCode	lastSeen
0	2016-03-24 00:00:00	0	70435	2016-04-07 03:16:57
1	2016-03-24 00:00:00	0	66954	2016-04-07 01:46:50
2	2016-03-14 00:00:00	0	90480	2016-04-05 12:47:46
3	2016-03-17 00:00:00	0	91074	2016-03-17 17:40:17
4	2016-03-31 00:00:00	0	60437	2016-04-06 10:17:21

- **Column: 20**
- **Row: 371539**

Technology used



Data Inspection

- Data exploration

- Numerical data
- dateCrawled/name/seller/offerType/price/abtest/vehicleType/yearOfRegistration/gearbox/powerPS/model/kilometer/monthOfRegistration/fuelType/brand/notRepairedDamage/dateCreated/nrOfPictures/postalCode/lastSeen

Data Inspection

- Data exploration

- **Numerical data - statistical description**

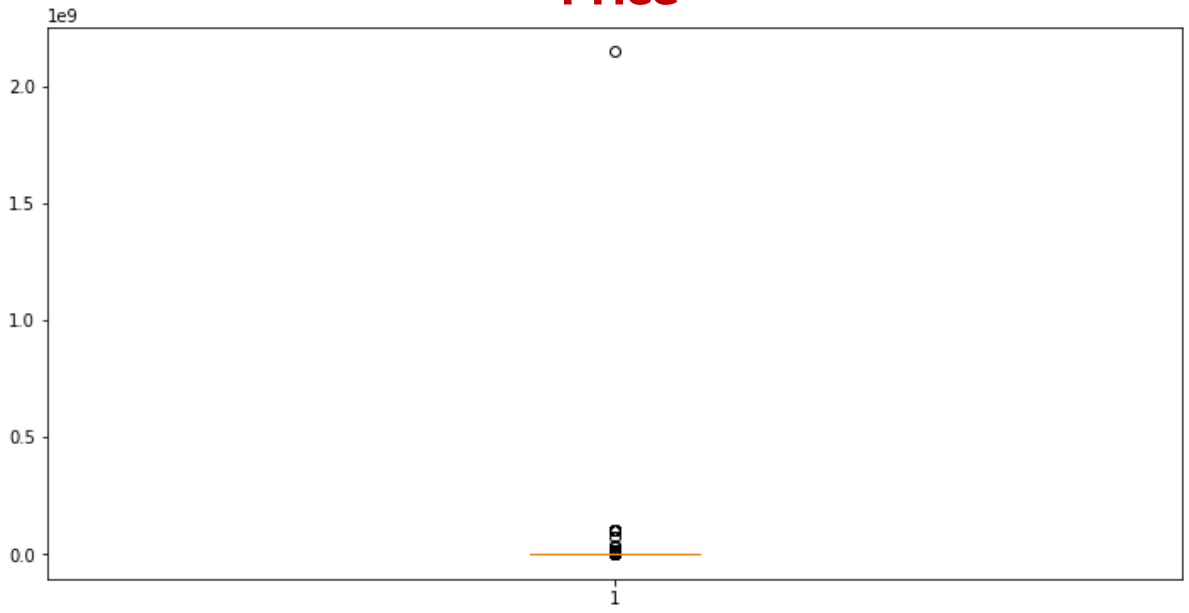
	price	yearOfRegistration	powerPS	kilometer	monthOfRegistration	nrOfPictures	postalCode
count	3.715280e+05	371528.000000	371528.000000	371528.000000	371528.000000	371528.0	371528.00000
mean	1.729514e+04	2004.577997	115.549477	125618.688228	5.734445	0.0	50820.66764
std	3.587954e+06	92.866598	192.139578	40112.337051	3.712412	0.0	25799.08247
min	0.000000e+00	1000.000000	0.000000	5000.000000	0.000000	0.0	1067.00000
25%	1.150000e+03	1999.000000	70.000000	125000.000000	3.000000	0.0	30459.00000
50%	2.950000e+03	2003.000000	105.000000	150000.000000	6.000000	0.0	49610.00000
75%	7.200000e+03	2008.000000	150.000000	150000.000000	9.000000	0.0	71546.00000
max	2.147484e+09	9999.000000	20000.000000	150000.000000	12.000000	0.0	99998.00000

Data Inspection – Data exploration

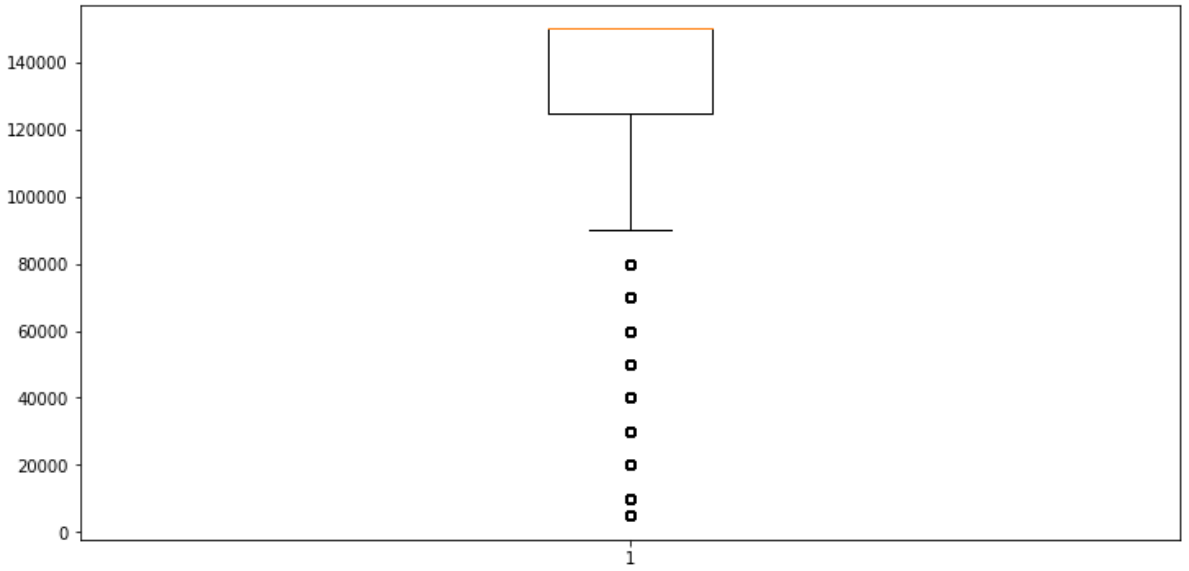
- Distribution

- Boxplot (numerical data) - outlier

Price



Kilometer

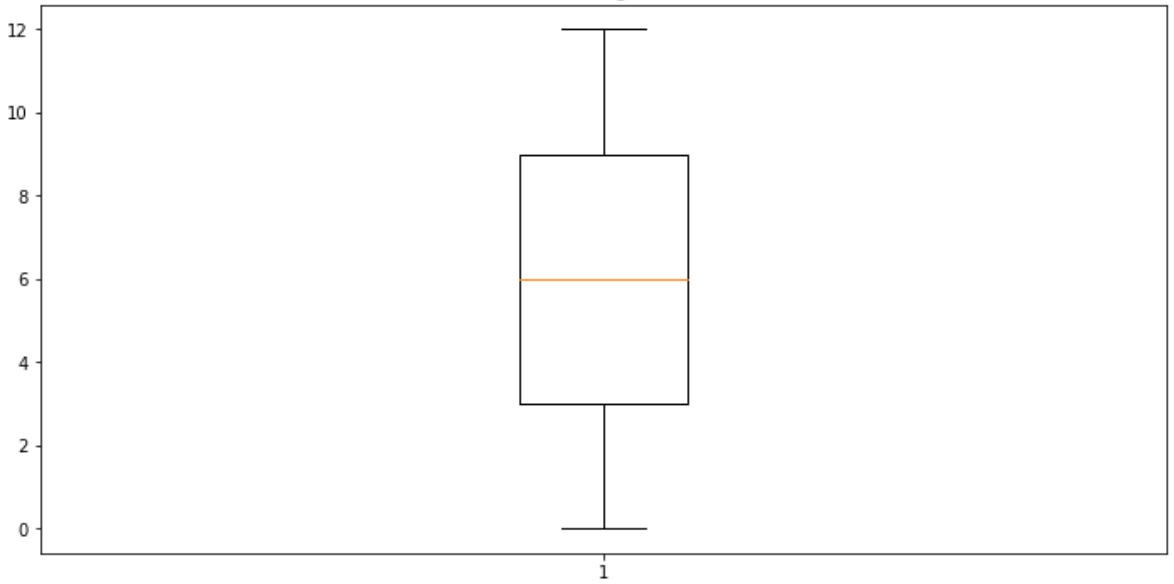


Data Inspection – Data exploration

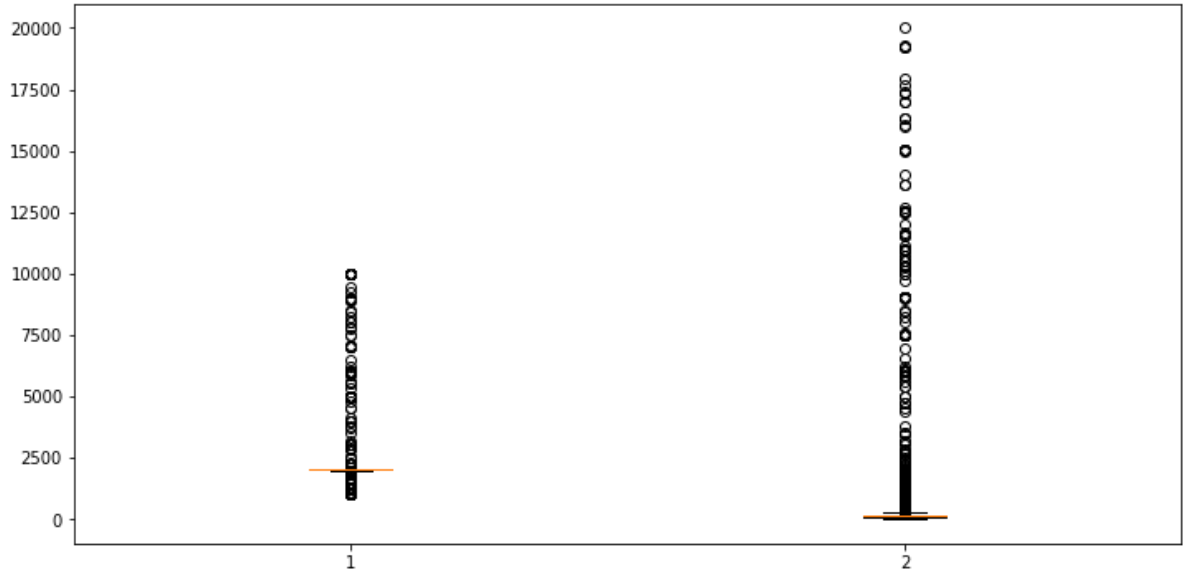
- Distribution

- Boxplot (numerical data) - outlier

monthOfRegistration



YearOfRegistration, powerPS

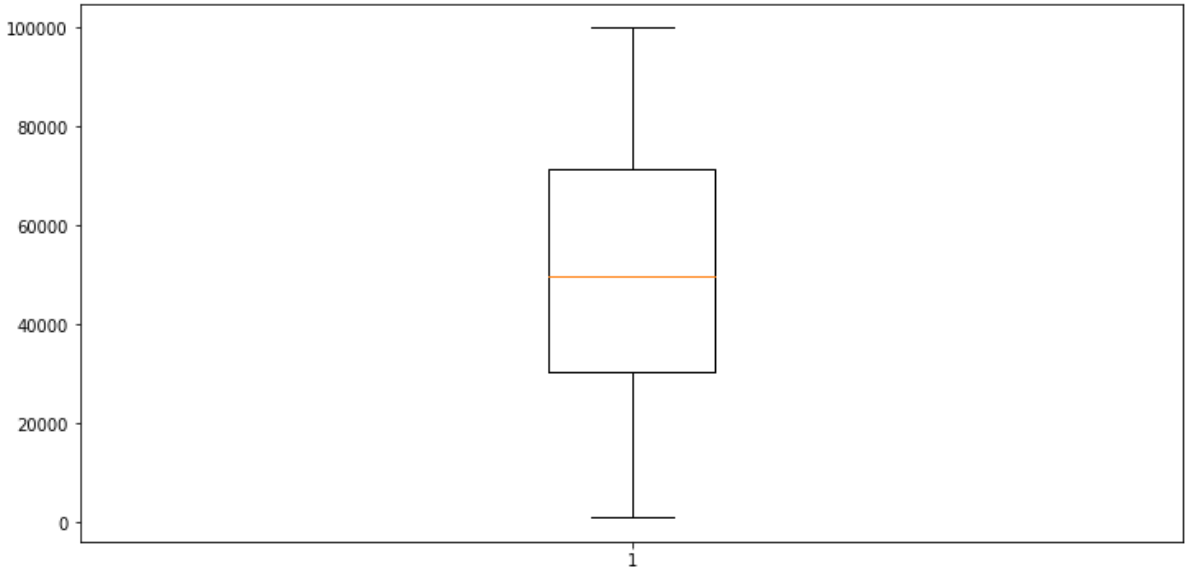


Data Inspection – Data exploration

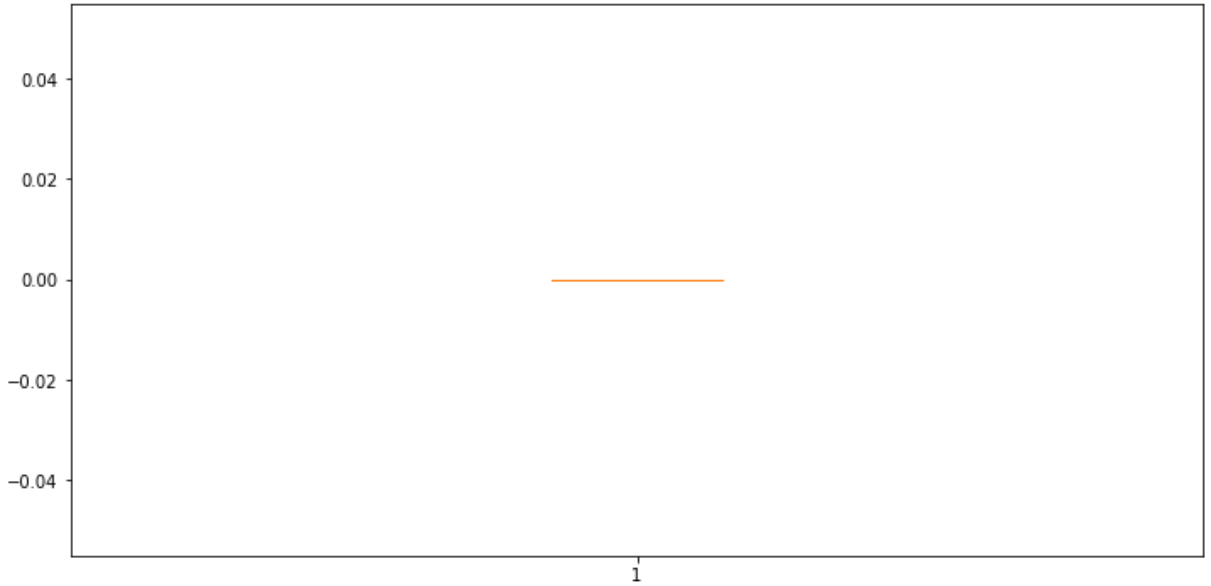
- Distribution

- Boxplot (numerical data) - outlier

PostalCode



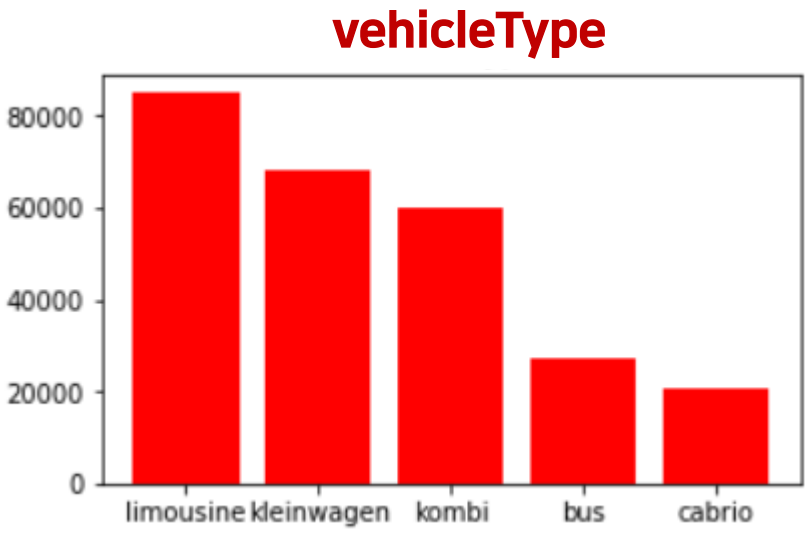
nrOfPictures

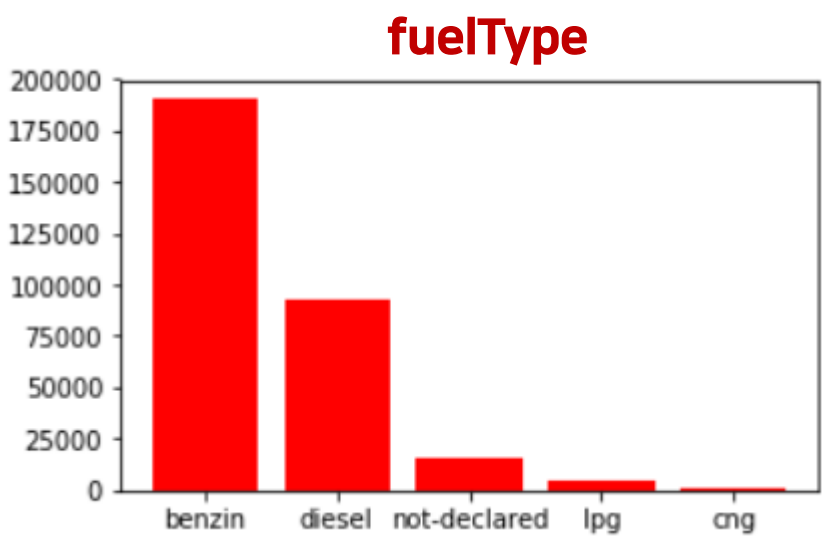


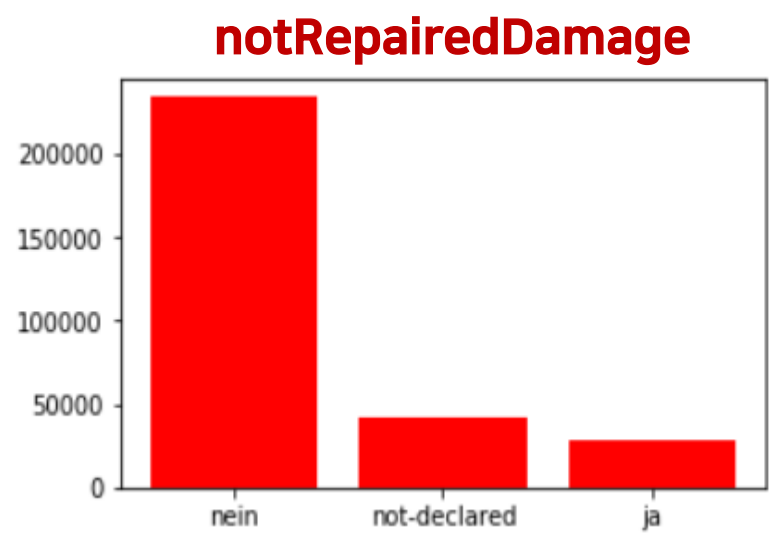
Data Inspection – Data exploration

- Distribution

- Histogram (Categories)



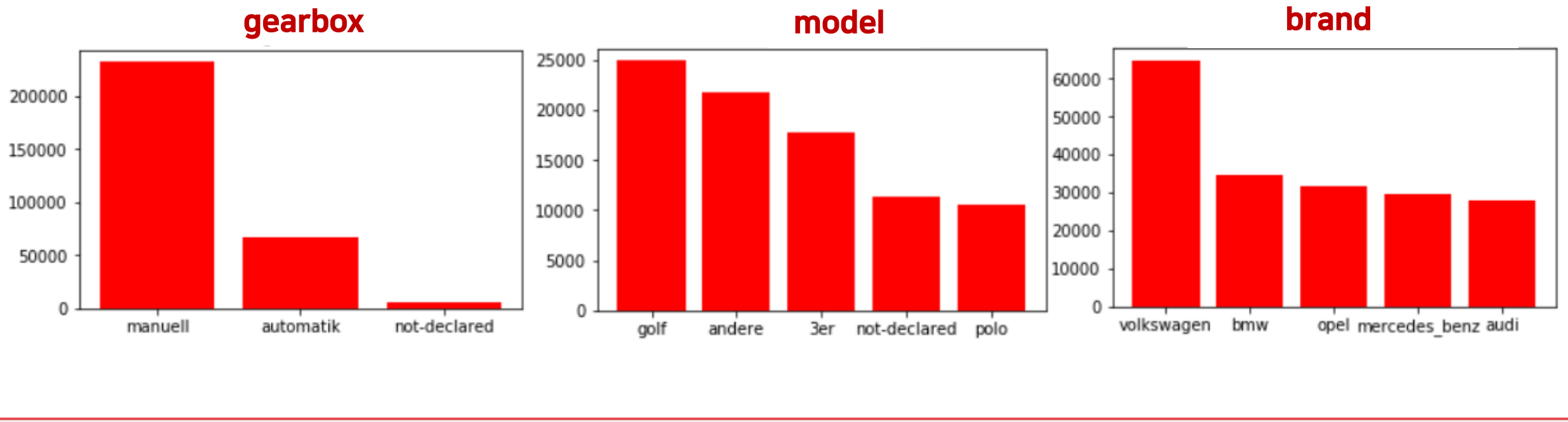




Data Inspection – Data exploration

- Distribution

- Histogram (Categories)



Data Preprocessing - Restructuring

- **Vertical decomposition**
  - Drop columns by two criteria

1) Columns with a data single value

   – Seller/offerType/abtest/nrOfPictures

2) Data crawling information

   – dataCrawled/lastSeen/postalCode/dateCreated

```
original.drop(['seller', 'offerType', 'abtest', 'dateCrawled', 'nrOfPictures', 'lastSeen', 'postalCode', 'dateCreated'],
              axis='columns', inplace=True)
```

Data Preprocessing – Data value changes

- **Cleaning dirty data**
  - Outlier – **delete**
  - yearOfRegistration/price/powerPS

```
data = data[
        (data.yearOfRegistration <= 2019) & (data.yearOfRegistration >= 1950)
      & (data.price >= 100) & (data.price <= 150000)
      & (data.powerPS >= 10) & (data.powerPS <= 500)]
```

Wrong data

Data Preprocessing – Data value changes

- **Cleaning dirty data**

  - **Unusable data – delete duplicate data**

```
data = original.drop_duplicates(['name','price','vehicleType','yearOfRegistration'
                                 ,'gearbox','powerPS','model','kilometer','monthOfRegistration'
                                 ,'notRepairedDamage'])
```

Data Preprocessing – Data value changes

- **Cleaning dirty data**
  - Missing data
    - fill mode of data
    - fill 'not declared'

Before

```
** Befo **
dateCrawled                0
name                       0
seller                     0
offerType                  0
price                      0
abtest                     0
vehicleType            30952
yearOfRegistration         0
gearbox                15976
powerPS                    0
model                      0
kilometer                  0
monthOfRegistration        0
fuelType               26092
brand                      0
notRepairedDamage      62892
dateCreated                0
nrOfPictures               0
postalCode                 0
lastSeen                   0
dtype: int64
```

- Mode of model

0	NaN
3	kleinwagen
9	kleinwagen
32	limousine
35	NaN
48	NaN
59	limousine
75	limousine
78	NaN
115	NaN
117	limousine
128	limousine
130	NaN
140	limousine
151	kleinwagen
163	limousine
169	limousine
195	bus

→

0	limousine
3	kleinwagen
9	kleinwagen
32	limousine
35	limousine
48	limousine
59	limousine
75	limousine
78	limousine
115	limousine
117	limousine
128	limousine
130	limousine
140	limousine
151	kleinwagen
163	limousine
169	limousine
195	bus

- fill 'not declared'

0	NaN
2	NaN
3	nein
4	nein
5	ja
6	nein
7	nein
8	NaN
9	NaN
10	nein

→

0	not-declared
2	not-declared
3	nein
4	nein
5	ja
6	nein
7	nein
8	not-declared
9	not-declared
10	nein

Data Preprocessing – Data value changes

- **Cleaning dirty data**
  - Missing data
    - fill mode of data
    - fill 'not declared'

- Mode of model

0	NaN
3	kleinwagen
9	kleinwagen
32	limousine
35	NaN
48	NaN
59	limousine
75	limousine
78	NaN
115	NaN
117	limousine
128	limousine
130	NaN
140	limousine
151	kleinwagen
163	limousine
169	limousine
195	bus

0	limousine
3	kleinwagen
9	kleinwagen
32	limousine
35	limousine
48	limousine
59	limousine
75	limousine
78	limousine
115	limousine
117	limousine
128	limousine
130	limousine
140	limousine
151	kleinwagen
163	limousine
169	limousine
195	bus

0	NaN
2	NaN
3	nein
4	nein
5	ja
6	nein
7	nein
8	NaN
9	NaN
10	nein

0	not-declared
2	not-declared
3	nein
4	nein
5	ja
6	nein
7	nein
8	not-declared
9	not-declared
10	nein

After

```
** After **
dateCrawled                0
name                       0
seller                     0
offerType                  0
price                      0
abtest                     0
vehicleType                0
yearOfRegistration         0
gearbox                    0
powerPS                    0
model                      0
kilometer                  0
monthOfRegistration        0
fuelType                   0
brand                      0
notRepairedDamage          0
dateCreated                0
nrOfPictures               0
postalCode                 0
lastSeen                   0
dtype: int64
```

Data Preprocessing – Data value changes

- **Cleaning dirty data**
  - **After cleaning dirty data**

```
** After **
name                 0
price                0
vehicleType          0
yearOfRegistration   0
gearbox              0
powerPS              0
model                0
kilometer            0
monthOfRegistration  0
fuelType             0
brand                0
notRepairedDamage    0
dtype: int64


** Size **
292786
```

Data Preprocessing – Data value changes

- **Data standardization**
  - **StandardScaler**

```
# standardization 수행
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_scale = scaler.fit_transform(X)
X_scale = pd.DataFrame(X_scale, columns = X.columns)
print(X_scale)
print(Y)
```

Data Preprocessing – Data value changes

- **Data standardization**
  - **StandardScaler**

yearOfRegistration	powerPS	kilometer	monthOfRegistration	gearbox_encode	notRepairedDamage_encode	model_encode	brand_encode	fuelType_encode	vehicleType_encode
1.155415	1.068103	-0.012251	-0.280884	0.457453	-2.630540	0.817615	-1.450419	0.639510	-0.850946
0.125466	0.618418	-0.012251	0.559298	-1.800507	-1.060282	0.230663	-0.482839	0.639510	1.908559
-0.315940	-0.847221	0.624848	-0.000823	0.457453	0.509977	0.216688	1.303463	-0.638061	-0.299045
0.714008	-0.947150	-0.904190	0.279238	0.457453	0.509977	0.007062	0.782459	0.639510	-0.299045
-1.198753	-0.397536	0.624848	1.119420	0.457453	-2.630540	-1.278644	-1.375990	-0.638061	0.804757
0.125466	-0.280951	0.624848	0.559298	0.457453	0.509977	-1.320569	0.335883	-0.638061	-1.402847
1.596821	-0.014471	-2.433227	0.559298	0.457453	-1.060282	-0.579891	-0.780556	-0.638061	-1.954747
-0.757347	-0.414191	0.624848	-1.681188	0.457453	-1.060282	0.216688	1.303463	3.194651	-0.299045
0.125466	-0.347571	0.624848	1.679542	0.457453	0.509977	-1.292619	-0.110693	-0.638061	0.804757
0.272602	0.235354	0.624848	1.679542	0.457453	-2.630540	0.971341	1.303463	0.639510	0.252856
-1.198753	-0.181021	0.624848	1.399481	0.457453	-1.060282	0.971341	1.303463	-0.638061	0.252856
0.125466	0.085459	0.624848	-1.121066	0.457453	0.509977	0.971341	1.303463	3.194651	0.252856
1.155415	1.068103	-1.413869	-0.841006	0.457453	0.509977	0.831590	0.187025	0.639510	1.908559
1.891092	-1.097045	0.624848	-1.681188	2.715412	-1.060282	1.013266	1.303463	-0.638061	1.356658
0.125466	-0.847221	0.624848	-1.121066	-1.800507	0.509977	1.753945	0.484742	-0.638061	-0.299045
0.566873	0.168734	0.624848	-0.000823	0.457453	0.509977	-0.579891	-0.780556	0.639510	-1.954747
0.125466	-0.397536	0.624848	-1.401127	0.457453	0.509977	-0.971193	-0.036263	-0.638061	-1.954747
0.861144	0.568453	-0.649350	-0.560945	0.457453	0.509977	1.376618	1.303463	-0.638061	-0.850946

Data Preprocessing – Feature engineering

- **Feature creation**
  - **LabelEncoder**

    convert string/integer labels to 0~K-1 integer

```
# columns that value is string
labels = ['gearbox', 'notRepairedDamage', 'model', 'brand', 'fuelType', 'vehicleType']
les = {}


# label encoder
# encode string columns to 0~k-1 labels
# and add to exist dataframe --> feature creation
for l in labels:
    les[l] = preprocessing.LabelEncoder()
    les[l].fit(data[l].astype(str))
    tr = les[l].transform(data[l].astype(str))
    data.loc[:, l + '_encode'] = pd.Series(tr, index=data

# extract only necessary columns to analysis
labeled = data[ ['price'
                 ,'yearOfRegistration'
                 ,'powerPS'
                 ,'kilometer'
                 ,'monthOfRegistration']
               + [x+"_encode" for x in labels]]
```

	gearbox_encode	notRepairedDamage_encode	model_encode	brand_encode	fuelType_encode	vehicleType_encode
1	1	0	161	1	3	3
2	0	1	119	14	3	8
3	1	2	118	38	1	4
4	1	2	103	31	3	4
5	1	0	11	2	1	6
6	1	2	8	25	1	2
8	1	1	61	10	1	1
9	1	1	118	38	7	4
10	1	2	10	19	1	6

Data Preprocessing – Feature engineering

- Feature reduction

- PCA: Remove unimportant columns

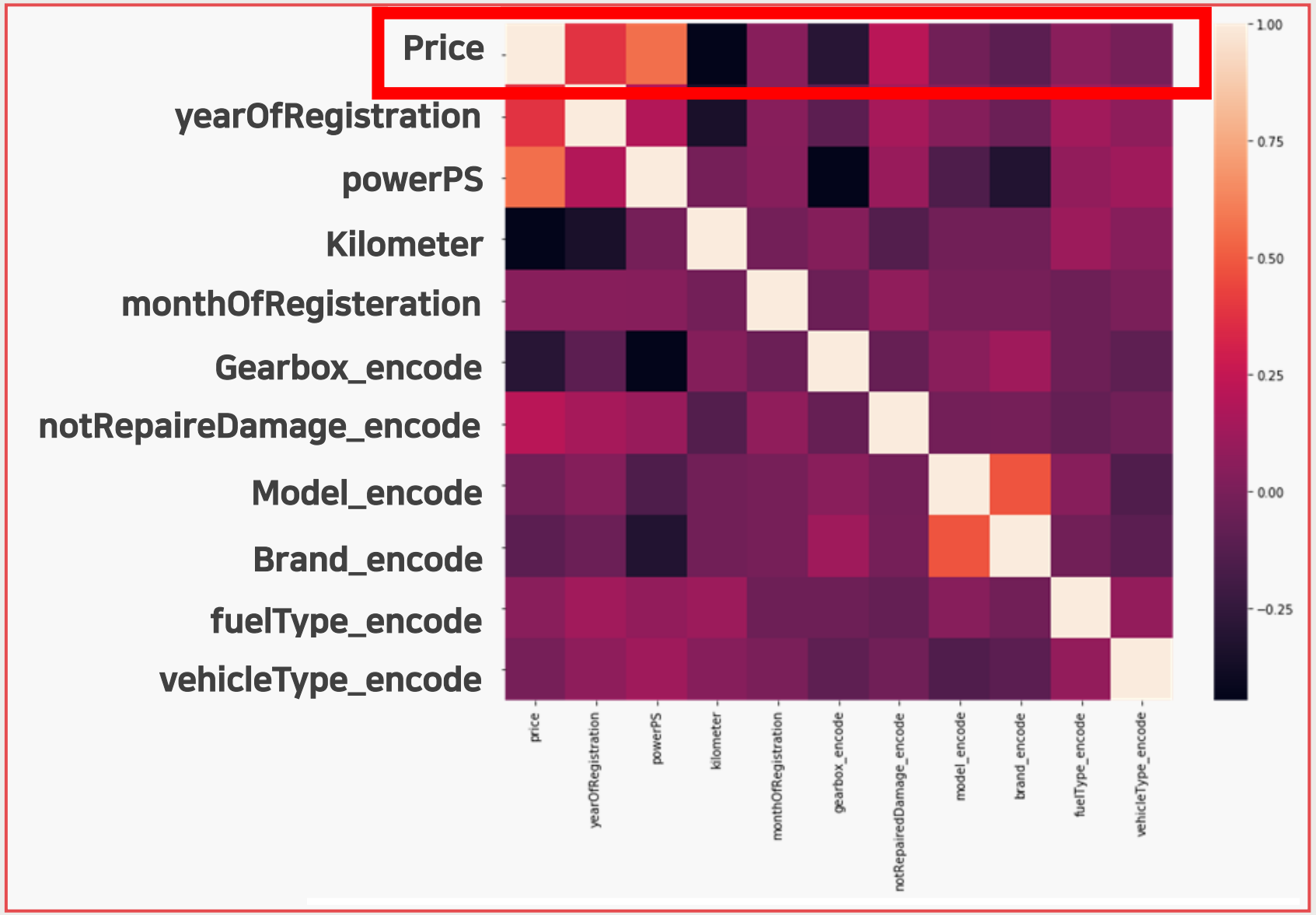
```
from sklearn.decomposition import PCA

pca = PCA(n_components=9)
pca.fit(X_scale)
print(pca)
#print(pca.fit_transform(X_scale))
X_pc = pca.fit_transform(X_scale)
```

```
PCA(copy=True, iterated_power='auto', n_components=9, random_state=None,
  svd_solver='auto', tol=0.0, whiten=False)
[[ 0.51320214  0.34765815  1.15574698 ...  2.31711877 -1.53252715
   0.71505555]
 [ 1.67950258  0.33484635  1.61618141 ...  1.00937559  0.44904199
   0.05727718]
 [-1.54939548 -0.1239111  -0.31337628 ... -0.65867368 -0.09206609
  -0.67350372]
 ...
 [-2.36046622 -0.47444293  0.89308125 ... -0.91325819  0.16019717
   0.47472059]
 [-1.25764329  0.41887714  1.16890561 ...  0.38043381  0.05824456
  -0.42891887]
 [ 2.75639705 -1.89194289 -0.54668956 ...  0.64262963 -1.68640807
   2.1617481 ]]
```

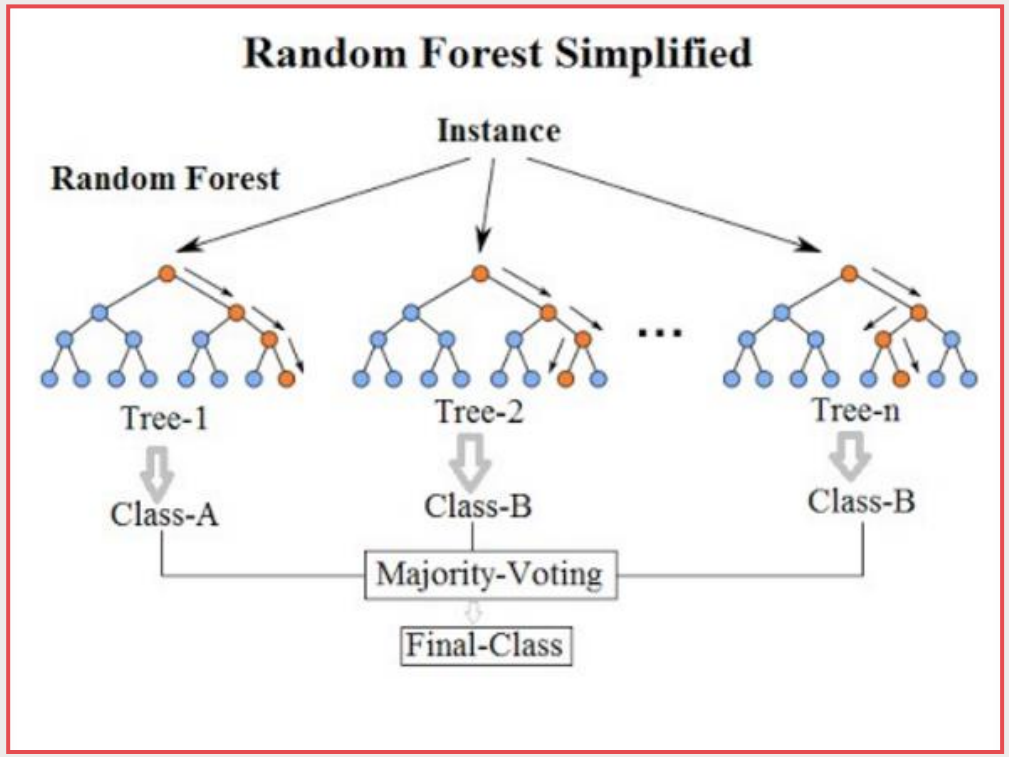
Data Analysis

- Correlation



Data Analysis

- **RandomForestRegressor**
  - Random Forest : a collection of Decision Trees



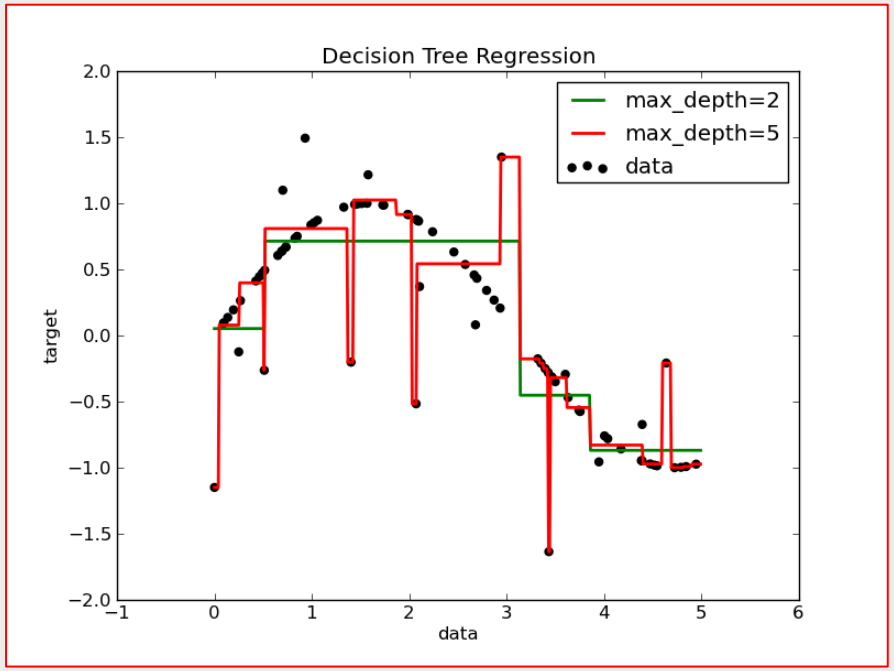
Data Analysis

- Decision Tree

```
from sklearn.tree import DecisionTreeRegressor


decisionTree = DecisionTreeRegressor().fit(X_train, y_train)
predictDecision = decisionTree.predict(X_test)
score = decisionTree.score(X_test, y_test)
print("Decision Tree Score : ",score)
```

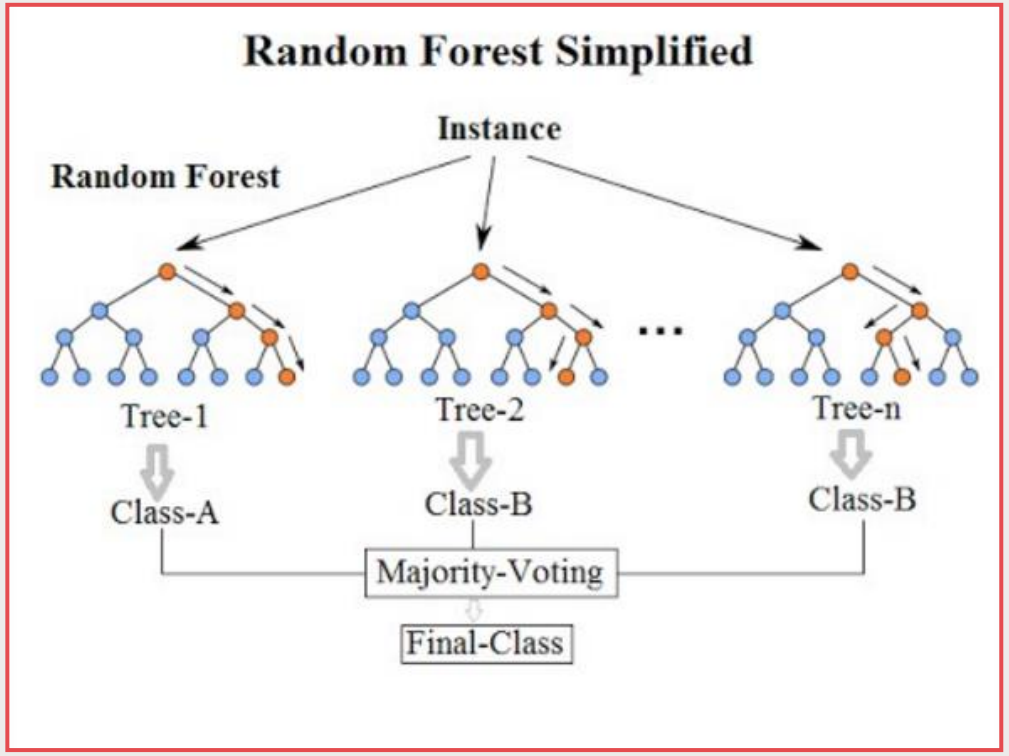
```
Decision Tree Score :  0.6670051955507951
```



The advantage of just using a typical decision tree is that it provides a clear process or result, but it is over-optimized for learning, making it easy to over-fit.

Data Analysis

- **RandomForestRegressor**
  - **Random Forest : a collection of Decision Trees**




```
Out-of-bag score estimate: 0.727
```

This accuracy was calculated by testing data that was not bagged when performing the random forest.

➔ This is provided by RandomForestRegressor.

Evaluation

- **MAPE (Mean Absolute Percentage Error)**
  - **Percentage error**

$$\mathrm{M} = \frac{100}{n} \sum_{t=1}^{n} \left| \frac{A_t - F_t}{A_t} \right|,$$

```
Accuracy using MAPE:
0.6620685610568278
```

  - **Divide the variation of the error into value**

➡ **compare it on the same basis on the ratio**

Evaluation

- **Accuracy too low**
  - **Back to the feature reduction process**

1. Objective Setting
2. Data Curation
3. Data Inspection
4. Data Preprocessing
5. Data Analysis
6. Evaluation
7. Deployment

→

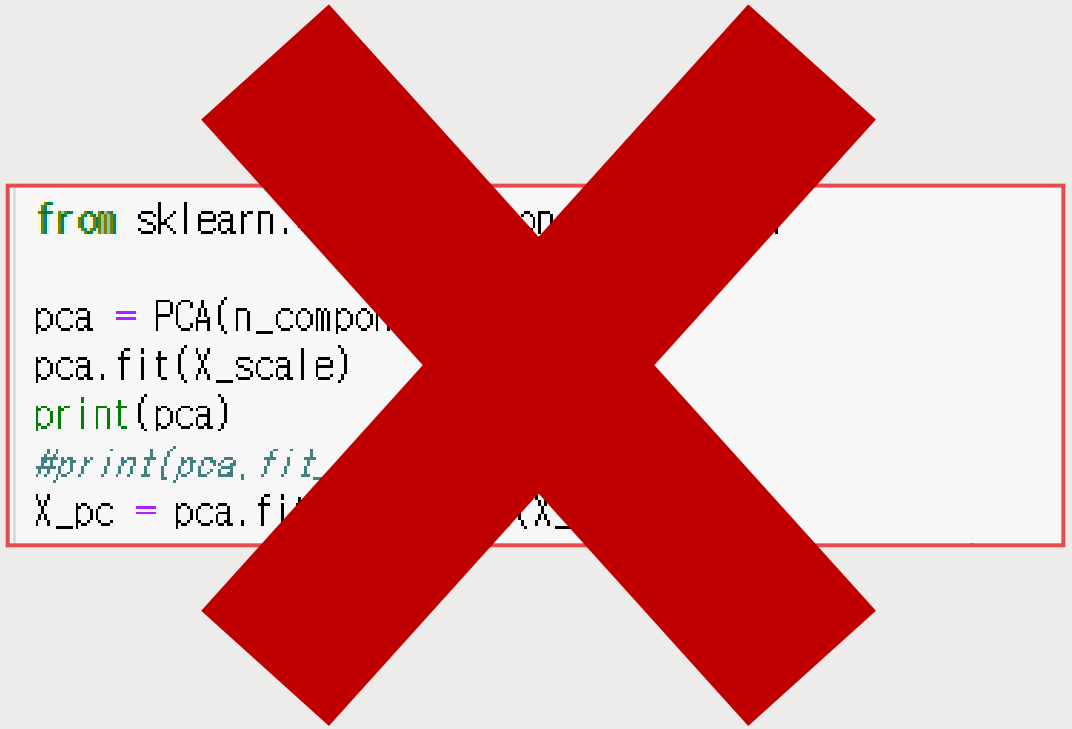
- Data Restructuring
- Data Value Changes
- Feature Engineering
- Data Reduction

→

- Feature Creation
- Feature Selection
- Feature Reduction

Data Preprocessing – Feature engineering

- **Feature reduction**
  - **PCA**



Data analysis again

Data Analysis

- **Random Forest Regressor**

Out-of-bag score estimate: 0.727

Out-of-bag score estimate: 0.881

Accuracy using MAPE:
0.6620685610568278

Accuracy using MAPE:
0.8038865814284948

Accuracy increase

```
Original:  2650
Predicted:  2654.568574562228

Original:  850
Predicted:  845.0911665290445

Original:  2650
Predicted:  2644.626602212917

Original:  700
Predicted:  706.1552333992726

Original:  699
Predicted:  706.0998480946428

Original:  1250
Predicted:  1257.3755100013957

Original:  3240
Predicted:  3247.4809935477892

Original:  4400
Predicted:  4408.258045045875
```

```
Original:  8200
Predicted:  7530.771931988155

Original:  3650
Predicted:  2980.579511481714

Original:  3300
Predicted:  2629.2927554221606

Original:  2299
Predicted:  1628.1638975428064

Original:  5000
Predicted:  5670.983583491222

Original:  900
Predicted:  1571.0988106675288
```

Predicted result

Conclusion

Two failures

- Decision tree
- PCA

We repeated data inspection and data preprocessing in this part.

As data is changed little by little, accuracy of the model has changed.

Conclusion

“Importance of **iterative** execution and **preprocessing**”

Thank you