Lab Assignment 8
Chaudhary Hamdan
1905387
Date: 23-03-2022

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.metrics import recall_score
from sklearn.metrics import precision_score
from sklearn.metrics import f1_score

df = pd.read_csv('car_sample.csv', encoding = "ISO-8859-1")
```

1.Find out if following variables are significant or insignificant and need to be dropped.

i) Seller-insignificant
ii) offerType-insignificant
iii)abtest-insignificant
Iv)vehicleType-significant
V)gearbox,
Vi)Model
Vii)Kilometer
Viii)Fueltype
Ix)Brand
X)notRepairedDamage

```
cols = ['dateCrawled', 'name', 'dateCreated', 'lastSeen']
df.drop(columns=cols, inplace=True)
df.head()
```

	seller	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthOfRegistratior
0	private	offer	4450	test	limousine	2003	manual	150	3er	150000	3
1	private	offer	13299	control	suv	2005	manual	163	xc_reihe	150000	6
2	private	offer	3200	test	bus	2003	manual	101	touran	150000	11
3	private	offer	4500	control	small car	2006	manual	86	ibiza	60000	12
4	private	offer	18750	test	suv	2008	automatic	185	xc_reihe	150000	11

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50001 entries, 0 to 50000
Data columns (total 15 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   seller               50001 non-null  object
 1   offerType            50001 non-null  object
 2   price                50001 non-null  int64
 3   abtest               50001 non-null  object
 4   vehicleType          44813 non-null  object
 5   yearOfRegistration   50001 non-null  int64
 6   gearbox              47177 non-null  object
 7   powerPS              50001 non-null  int64
 8   model                47243 non-null  object
 9   kilometer            50001 non-null  int64
 10  monthOfRegistration  50001 non-null  int64
 11  fuelType             45498 non-null  object
 12  brand                50001 non-null  object
 13  notRepairedDamage    40285 non-null  object
 14  postalCode           50001 non-null  int64
dtypes: int64(6), object(9)
memory usage: 5.7+ MB
```

```
for col in df.columns:
    print(col)
    print(df[col].value_counts())
    print()
```

```
seller
private       49999
commercial        2
Name: seller, dtype: int64

offerType
offer      49998
request        3
Name: offerType, dtype: int64

price
0        1451
500       742
1500      705
1000      647
2500      594
         ...
18181       1
955         1
2970        1
40830       1
15880       1
Name: price, Length: 2393, dtype: int64
```

```
cols = ['seller', 'offerType', 'notRepairedDamage']
df.drop(columns=cols, inplace=True)
df.head()

cols = ['abtest', 'vehicleType', 'gearbox', 'model', 'fuelType', 'brand']

for col in cols:
    le = preprocessing.LabelEncoder()
    df[col] = le.fit_transform(df[col].astype(str))
df.head()
```

	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthOfRegistration	fuelType	brand
0	4450	1	3	2003	1	150	11	150000	3	1	2
1	13299	0	8	2005	1	163	243	150000	6	1	39
2	3200	1	0	2003	1	101	221	150000	11	1	38
3	4500	0	6	2006	1	86	120	60000	12	7	30
4	18750	1	8	2008	0	185	243	150000	11	1	39

df.corr()

	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthOfRegist
price	1.000000	0.002790	-0.011208	0.017604	-0.018165	0.020429	-0.002403	-0.045458	0.000582
abtest	0.002790	1.000000	0.005034	0.003324	-0.003996	0.001375	-0.001415	-0.003027	0.000621
vehicleType	-0.011208	0.005034	1.000000	0.000573	-0.000225	-0.035590	-0.037315	0.020446	0.006124
yearOfRegistration	0.017604	0.003324	0.000573	1.000000	0.029205	-0.004394	0.008299	-0.064188	-0.023152
gearbox	-0.018165	-0.003996	-0.000225	0.029205	1.000000	-0.142459	0.046735	0.005481	-0.123792
powerPS	0.020429	0.001375	-0.035590	-0.004394	-0.142459	1.000000	-0.035191	-0.016447	0.034345
model	-0.002403	-0.001415	-0.037315	0.008299	0.046735	-0.035191	1.000000	-0.043010	-0.028372
kilometer	-0.045458	-0.003027	0.020446	-0.064188	0.005481	-0.016447	-0.043010	1.000000	0.001985
monthOfRegistration	0.000582	0.000621	0.006124	-0.023152	-0.123792	0.034345	-0.028372	0.001985	1.000000
fuelType	-0.013127	0.004686	-0.035184	-0.012598	0.126904	-0.044093	-0.034566	-0.104424	-0.062377
brand	-0.007697	0.006246	0.012066	0.004461	0.120576	-0.083801	0.435585	-0.031284	-0.018635
postalCode	0.005916	0.003096	-0.013254	-0.001615	0.003200	0.017415	-0.051870	-0.024076	0.019050

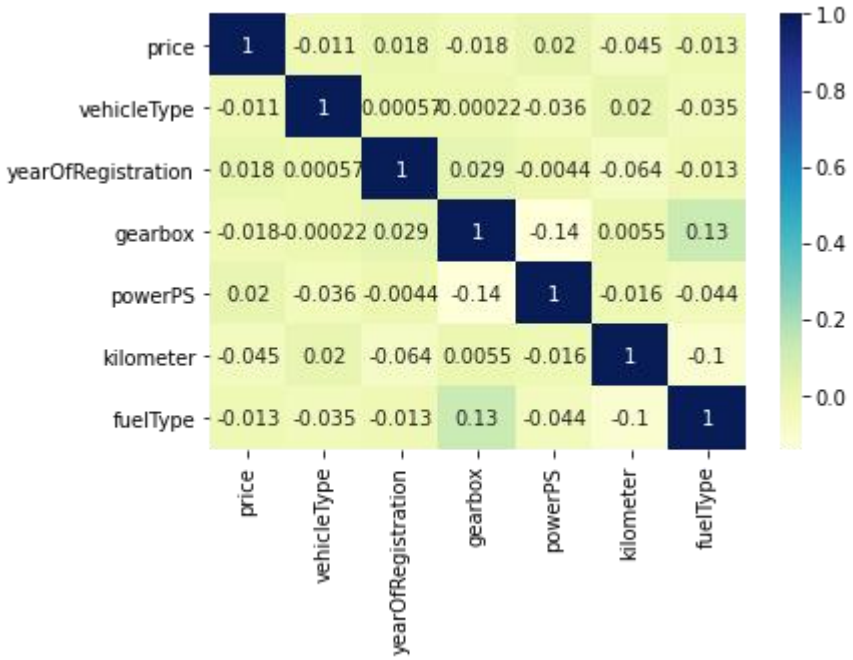
2. Drop insignificant variables from dataframe ‘cars’

```
cols = ['abtest', 'model', 'monthOfRegistration', 'brand', 'postalCode']
df.drop(columns=cols, inplace=True)
df.head()
```

	price	vehicleType	yearOfRegistration	gearbox	powerPS	kilometer	fuelType
0	4450	3	2003	1	150	150000	1
1	13299	8	2005	1	163	150000	1
2	3200	0	2003	1	101	150000	1
3	4500	6	2006	1	86	60000	7
4	18750	8	2008	0	185	150000	1

3. Find correlation between all numerical variables and find which variable has the highest correlation with price

```
cor = df.corr()
sns.heatmap(cor, cmap="YlGnBu", annot=True)
plt.show()
```




```
print(cor['price'])
print('Highest: kilometer (abs vale of 0.045458)')
```

```
price                1.000000
vehicleType         -0.011208
yearOfRegistration   0.017604
gearbox             -0.018165
powerPS              0.020429
kilometer           -0.045458
fuelType            -0.013127
Name: price, dtype: float64
Highest: kilometer (abs vale of 0.045458)
```

4. Calculate the training data and testing data score using a linear regression model.

```
x_train, x_test, y_train, y_test = train_test_split(df.drop(columns = ['price']), df['price'], test_size = 0.2)
x_train.shape, y_train.shape, x_test.shape, y_test.shape

algo = "Linear Regression\n"
model = LinearRegression()
model.fit(x_train, y_train)
print(algo)

print('Training error')
y_pred = model.predict(x_train)
e = (y_pred - y_train)
e = e.dot(e)
e /= y_test.shape[0]
e = e**0.5
print(e)

print('Testing error')
y_pred = model.predict(x_test)
e = (y_pred - y_test)
e = e.dot(e)
e /= y_test.shape[0]
e = e**0.5
print(e)
```

```
Linear Regression

Training error
189127.72277489284
Testing error
30656.97146180956
```