Used_Car_Marketing_Analysis

KNN_Regression,Linear Regression,Random Forest

Download Dataset

https://www.kaggle.com/orgesleka/used-cars-database/discussion

Import Libraries and Load Dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_i

Enter your authorization code:
··········
Mounted at /content/drive
```

```
#Load libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
# Load dataset
df=pd.read_csv('../content/drive/My Drive/DSProject/Germany_Used_Car
```

```
/usr/local/lib/python3.6/dist-packages/IPython/core/interactiveshell.py:2718: D
  interactivity=interactivity, compiler=compiler, result=result)
```

Exploratory Analysis data

```
# Dataset preview
df.head()
```

	dateCrawled	name	price	vehicleType
0	3/21/2016 21:46	Opel_Corsa_1.2_16V___2_HAND__KLIMA__8_FACH_T?V...	2238	small car
1	4/4/2016 23:48	Mercedes_Benz_E_220_T_CDI_Avantgarde	12500	station wagon
2	3/17/2016 0:46	BMW_325_xi_E92_Coupe__EZ_12/2006___6__Gang_Sch...	13299	coupe
3	3/29/2016 18:51	BMW_520d_Touring_Xenon_Navi+_PCD_Sport_Comf._1...	17200	station wagon
4	3/28/2016 16:45	BMW_635_CSI_Schaltgetriebe_original_Fahrzeug_2...	8000	coupe

```
# Review all column index
df.columns.values
```

```
array(['dateCrawled', 'name', 'price', 'vehicleType',
       'yearOfRegistration', 'yearOfUsing', 'gearbox', 'powerPS', 'model',
       'kilometer', 'monthOfRegistration', 'fuelType', 'brand',
       'notRepairedDamage', 'dateCreated', 'postalCode', 'lastSeen'],
      dtype=object)
```

```
# Number of rows and columns
print(df.shape)
```

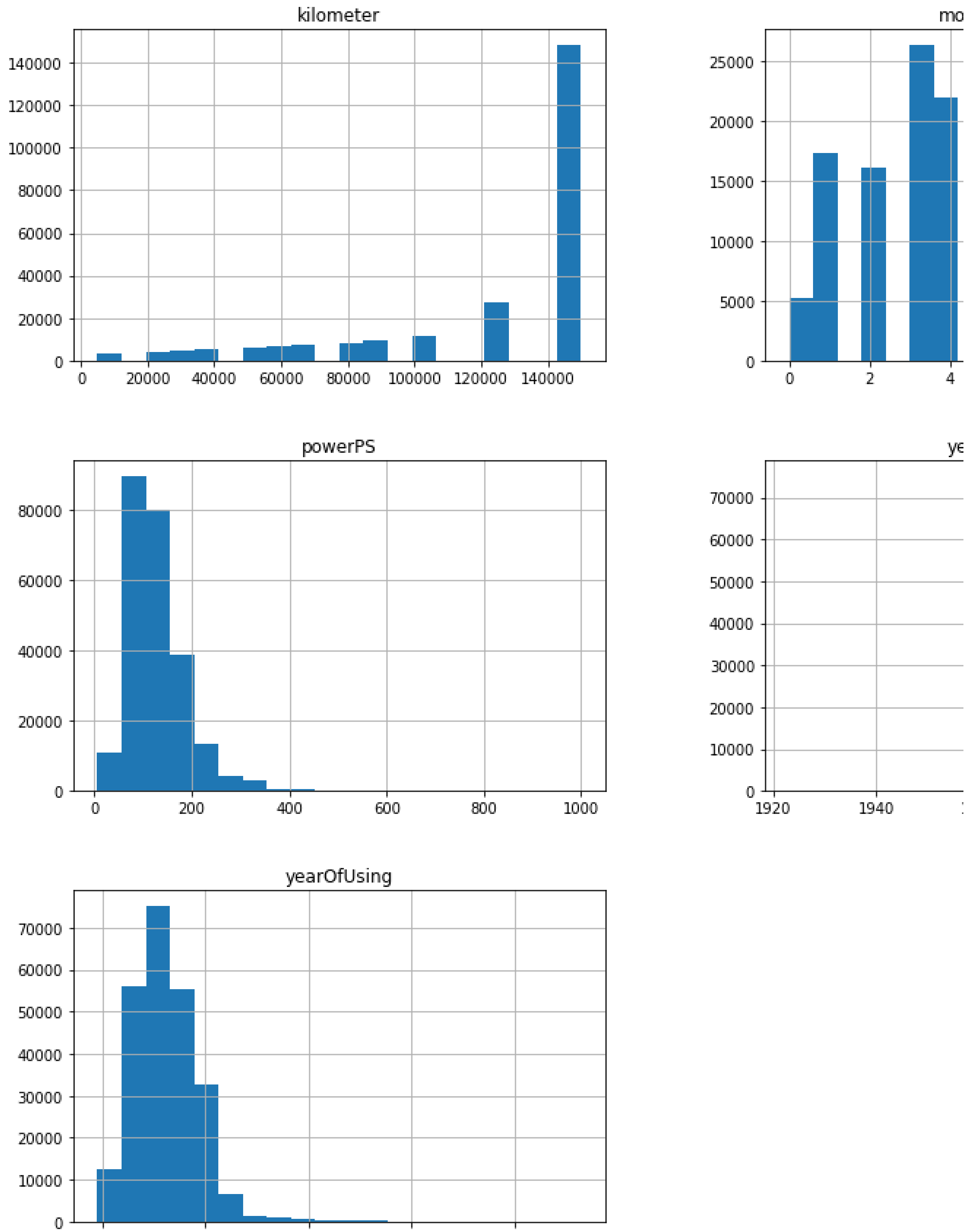
```
(242109, 17)
```

```
# Data types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 242109 entries, 0 to 242108
Data columns (total 17 columns):
dateCrawled            242109 non-null object
name                   242109 non-null object
price                  242109 non-null object
vehicleType            242109 non-null object
yearOfRegistration     242109 non-null int64
yearOfUsing            242109 non-null int64
gearbox                242109 non-null object
powerPS                242109 non-null int64
model                  242109 non-null object
kilometer              242109 non-null int64
monthOfRegistration    242109 non-null int64
fuelType               242109 non-null object
brand                  242109 non-null object
notRepairedDamage      242109 non-null object
dateCreated            242109 non-null object
postalCode             242109 non-null object
lastSeen               242109 non-null object
dtypes: int64(5), object(12)
memory usage: 31.4+ MB
```

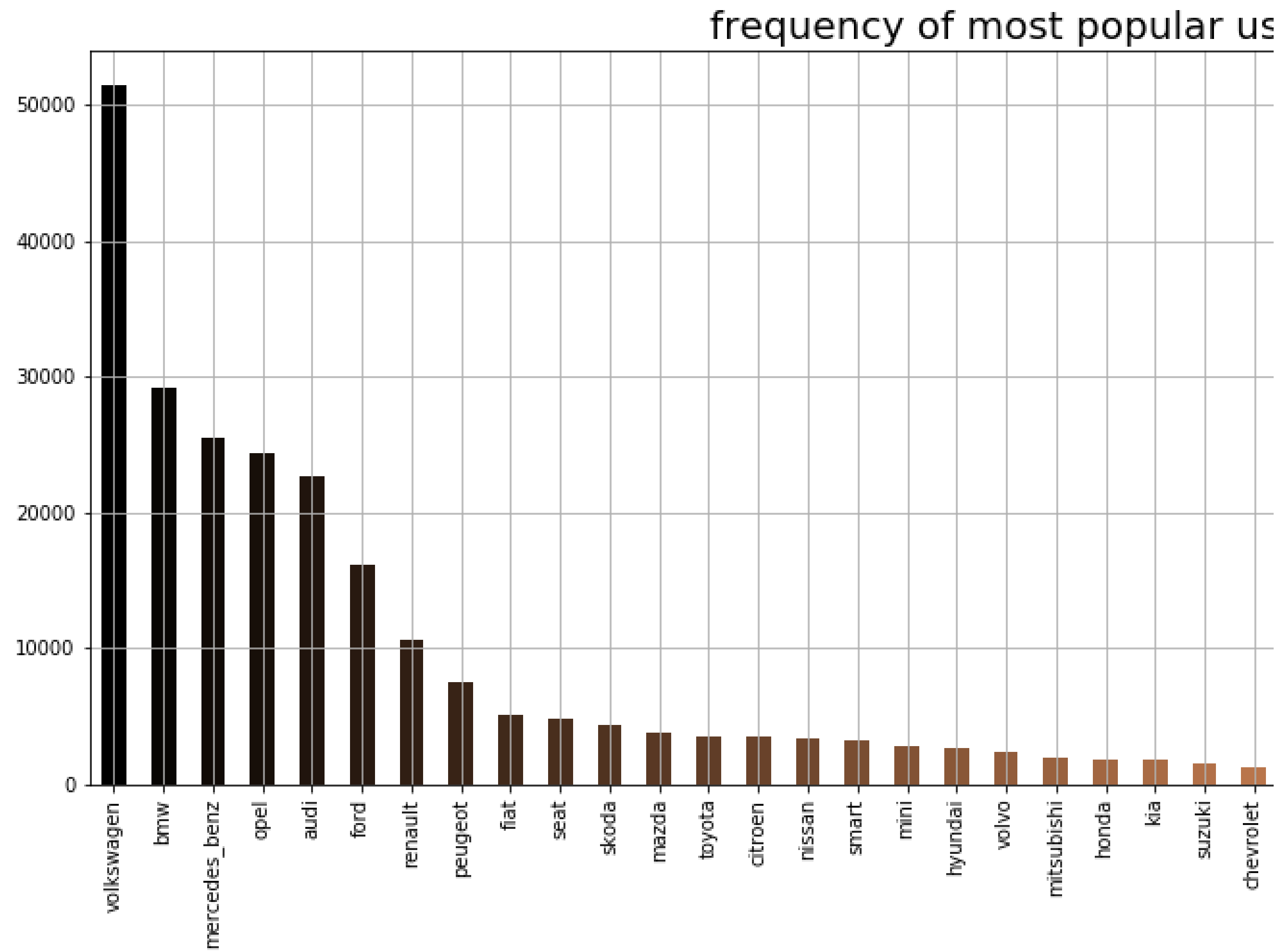
```
# Visualization for data statistic distribution
df.hist(figsize = (15,15),bins=20)
```

```
array([[<matplotlib.axes._subplots.AxesSubplot object at 0x7f1108ed9710>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f1108eb1c50>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f1108e71240>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f1108e217f0>],
       [<matplotlib.axes._subplots.AxesSubplot object at 0x7f1108dd1da0>,
        <matplotlib.axes._subplots.AxesSubplot object at 0x7f1108d8c390>]],
      dtype=object)
```



0 20 40 60 80

```
# Looking at the frequency of most popular Car Brand
plt.rcParams['figure.figsize'] = (18, 7)
color = plt.cm.copper(np.linspace(0, 1, 40))
df.brand.value_counts().head(40).plot.bar(color = color)
plt.title('frequency of most popular used car', fontsize = 20)
plt.xticks(rotation = 90 )
plt.grid()
plt.show()
```



Feature Engineering

```
# Review all column index
df.columns.values
```

```
array(['dateCrawled', 'name', 'price', 'vehicleType',
       'yearOfRegistration', 'yearOfUsing', 'gearbox', 'powerPS', 'model',
       'kilometer', 'monthOfRegistration', 'fuelType', 'brand',
       'notRepairedDamage', 'dateCreated', 'postalCode', 'lastSeen'],
      dtype=object)
```

```
# Check null data
df.isnull().sum()
```

```
dateCrawled            0
name                   0
price                  0
vehicleType            0
yearOfRegistration     0
yearOfUsing            0
gearbox                0
powerPS                0
model                  0
kilometer              0
monthOfRegistration    0
fuelType               0
brand                  0
notRepairedDamage      0
dateCreated            0
postalCode             0
lastSeen               0
dtype: int64
```

```
# We found 'price' is mixed data types with string ('Lower Saxony')
# Only four rows has 'Lower Saxony' so that we drop them.
df_remove = df[df.price == 'Lower Saxony']
df= df.drop(df_remove.index)
```

```
# Convert price to integer
df['price']=df['price'].astype(int)

# Convert to str
columns = ['yearOfRegistration','monthOfRegistration','postalCode']
for x in columns:
  df[x]=df[x].astype(str)
```

```
# Data Statistics
df.describe()
```

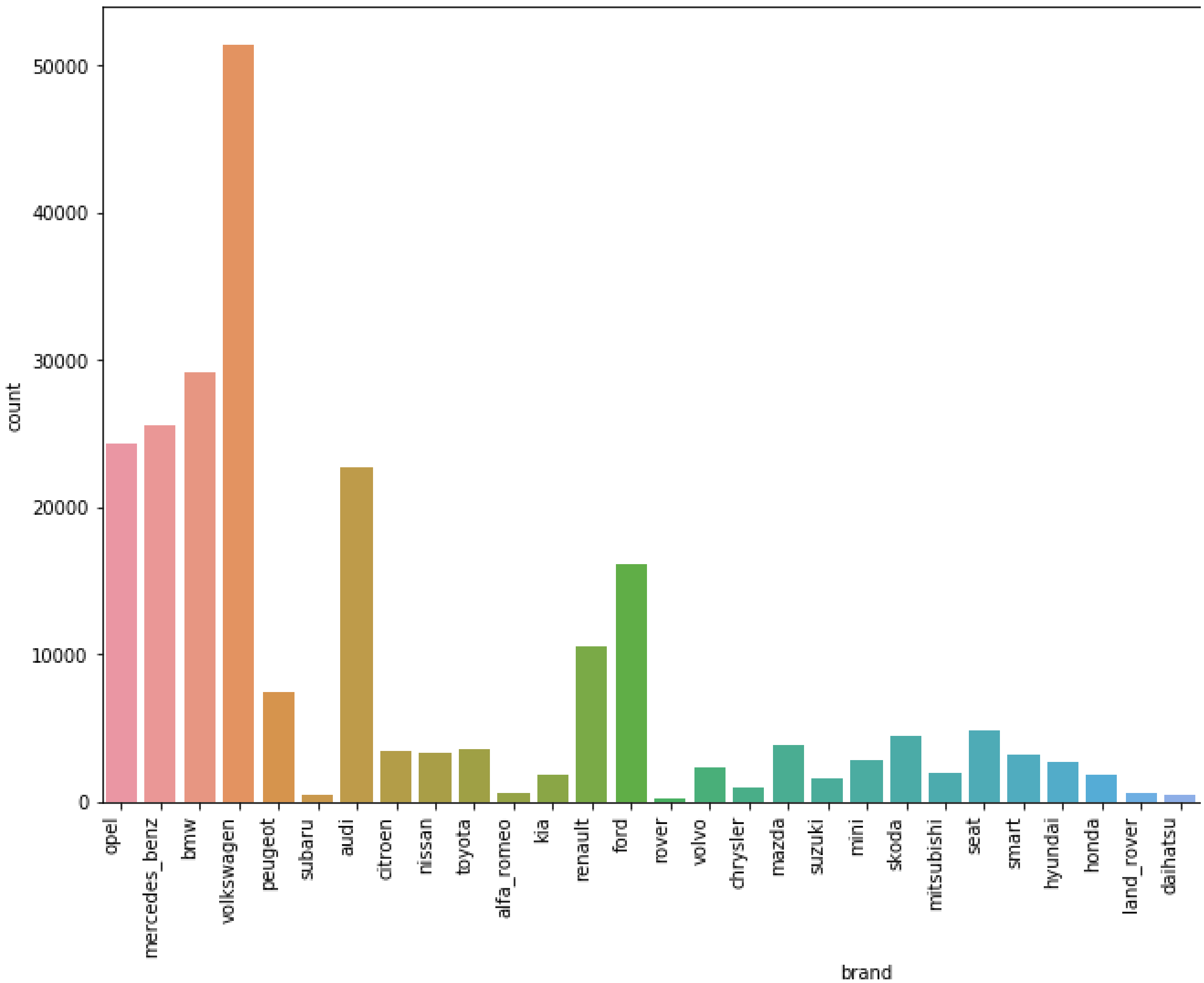
	price	yearOfUsing	powerPS	kilometer
count	242105.000000	242105.000000	242105.000000	242105.000000
mean	6752.196968	12.385717	129.297173	123789.264988
std	7656.033019	6.297437	61.339541	39741.321291
min	200.000000	-1.000000	6.000000	5000.000000
25%	1700.000000	8.000000	86.000000	100000.000000
50%	4000.000000	12.000000	116.000000	150000.000000
75%	8950.000000	16.000000	160.000000	150000.000000
max	99999.000000	93.000000	1000.000000	150000.000000

brand

```
df.brand.describe()
```

```
count         242105
unique            39
top       volkswagen
freq           51443
Name: brand, dtype: object
```

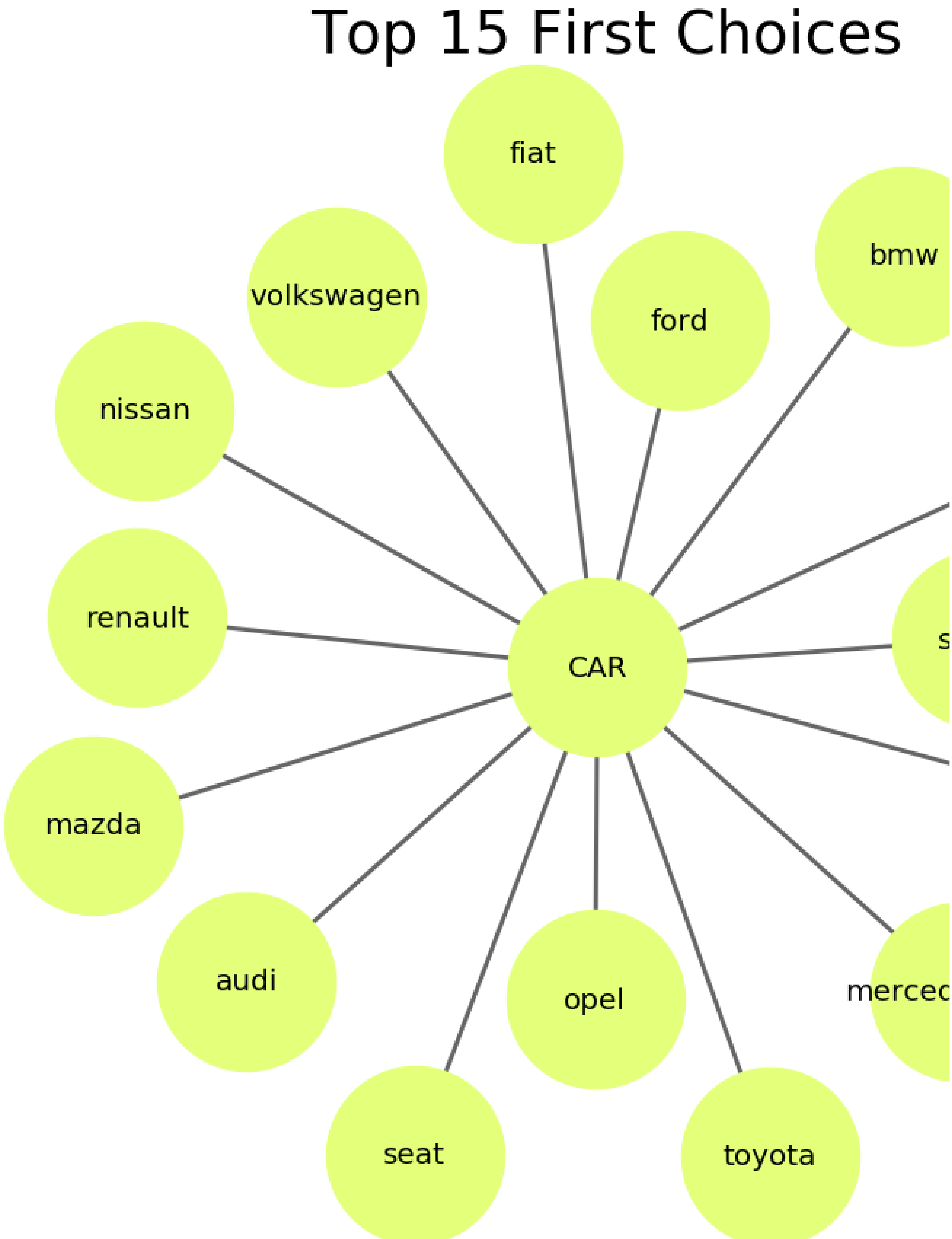
```
# Visualising all the brand
plt.figure(figsize=(15,8))
ax = sns.countplot(df.brand)
ax.set_xticklabels(ax.get_xticklabels(), rotation=90, ha="right")
plt.show()
```




```
y = df.brand.value_counts().to_frame()
y.head(10)
```

	brand
volkswagen	51443
bmw	29175
mercedes_benz	25526
opel	24321
audi	22699
ford	16094
renault	10546
peugeot	7412
fiat	5093
seat	4758

```
 Visualization for Brand
read new data frame brand1, will show top 15 best sale used car
and1=df.brand.value_counts().head(15).reset_index().rename(columns={



brand1['Car'] = 'CAR'
Car = brand1.truncate(before = -1, after = 15)

Car
import networkx as nx

Car = nx.from_pandas_edgelist(Car, source = 'Car', target = 'Brand',


# Visualising the top 15 brand
import warnings
warnings.filterwarnings('ignore')

plt.rcParams['figure.figsize'] = (15, 15)
pos = nx.spring_layout(Car)
color = plt.cm.Wistia(np.linspace(0, 15, 1))
nx.draw_networkx_nodes(Car, pos, node_size = 15000, node_color = col
nx.draw_networkx_edges(Car, pos, width = 3, alpha = 0.6, edge_color
nx.draw_networkx_labels(Car, pos, font_size = 20, font_family = 'sar
plt.axis('off')
plt.grid()
plt.title('Top 15 First Choices', fontsize = 40)
plt.show()
```




```
# Correlation table,feature and feature correlation
corMat = df.corr(method='pearson')
```

```
corMat
```

	price	yearOfUsing	powerPS	kilometer
price	1.000000	-0.504308	0.581624	-0.484205
yearOfUsing	-0.504308	1.000000	-0.211585	0.424714
powerPS	0.581624	-0.211585	1.000000	-0.013872
kilometer	-0.484205	0.424714	-0.013872	1.000000

```
# Heat map_positive and negative correlation
plt.figure(figsize=(8,4))
sns.heatmap(corMat, annot=True,cmap="terrain", )
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f11068a3278>
```

clean data

```
# Review all column index
df.columns.values
```

```
array(['dateCrawled', 'name', 'price', 'vehicleType',
       'yearOfRegistration', 'yearOfUsing', 'gearbox', 'powerPS', 'model',
       'kilometer', 'monthOfRegistration', 'fuelType', 'brand',
       'notRepairedDamage', 'dateCreated', 'postalCode', 'lastSeen'],
      dtype=object)
```

```
# Drop not important columns
df_new = df.drop(['dateCrawled', 'name', 'yearOfRegistration', 'mont
df_new.columns.values
```

```
array(['price', 'vehicleType', 'yearOfUsing', 'gearbox', 'powerPS',
       'model', 'kilometer', 'fuelType', 'brand', 'notRepairedDamage',
       'postalCode'], dtype=object)
```

```
# Label Encoding
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
columns = ['vehicleType', 'gearbox', 'model', 'fuelType', 'brand', '
le = LabelEncoder()
df_new[columns] = df_new[columns].apply(lambda x: le.fit_transform(x

df_new.head()
```

	price	vehicleType	yearOfUsing	gearbox	powerPS	model	kilometer	fuelType
0	2238	5	13	1	75	69	125000	6
1	12500	6	9	1	170	82	150000	1
2	13299	2	10	1	218	5	125000	6
3	17200	6	4	1	184	8	150000	1
4	8000	2	36	1	218	10	150000	6

```
df_new.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 242105 entries, 0 to 242108
Data columns (total 11 columns):
price                242105 non-null int64
vehicleType          242105 non-null int64
yearOfUsing          242105 non-null int64
gearbox              242105 non-null int64
powerPS              242105 non-null int64
model                242105 non-null int64
kilometer            242105 non-null int64
fuelType             242105 non-null int64
brand                242105 non-null int64
notRepairedDamage    242105 non-null int64
postalCode           242105 non-null int64
dtypes: int64(11)
memory usage: 32.2 MB
```

Split Dataset

```
from sklearn import neighbors
```

```
from sklearn.metrics import mean_squared_error
from math import sqrt


from sklearn.model_selection import train_test_split
#feature
X= df_new.drop(['price'], axis=1)
#lable
y =df_new['price']

X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=(
```

KNN_Regression

```
 Checnk K from 1 to 20
ıse_val = []
›r K in range(20):
  K = K+1
  model = neighbors.KNeighborsRegressor(n_neighbors = K)
  model.fit(X_train, y_train)
  y_pred=model.predict(X_test)
  error = sqrt(mean_squared_error(y_test,y_pred))
  rmse_val.append(error)
  #RootMeanSquareError get avarage error
  print('RMSE value for k= ' , K , 'is:', error)

/hen k = which number we can get best modle
·int('\nWhen k=',rmse_val.index(min(rmse_val))+1, 'we can get minmum
```

```
RMSE value for k=  1 is: 4979.083620730316
RMSE value for k=  2 is: 4639.4695260430935
RMSE value for k=  3 is: 4511.547261620518
RMSE value for k=  4 is: 4457.800532148259
RMSE value for k=  5 is: 4446.4365010197225
RMSE value for k=  6 is: 4446.592946786112
RMSE value for k=  7 is: 4444.9174576671785
RMSE value for k=  8 is: 4444.5148207438915
RMSE value for k=  9 is: 4447.485179544738
RMSE value for k=  10 is: 4455.54721489346
RMSE value for k=  11 is: 4467.10948983663
RMSE value for k=  12 is: 4479.778886598858
RMSE value for k=  13 is: 4495.478204768784
RMSE value for k=  14 is: 4506.629204576699
RMSE value for k=  15 is: 4518.263685895295
RMSE value for k=  16 is: 4532.454073668732
RMSE value for k=  17 is: 4548.850189172047
RMSE value for k=  18 is: 4561.336844808742
RMSE value for k=  19 is: 4572.244096040852
RMSE value for k=  20 is: 4582.587956341021

When k= 8 we can get minmum RMSE: 4444.5148207438915
```

Linear Regression

```
mport numpy as np
mport pandas as pd
mport matplotlib as mpl
mport matplotlib.pyplot as plt
lt.figure
mport seaborn as sns

rom sklearn.linear_model import LinearRegression
rom sklearn.metrics import mean_squared_error
rom math import sqrt

feature
= df_new.drop(['price'], axis=1)
lable
 =df_new['price']

_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.


egressor=LinearRegression()
egressor.fit(X_train,y_train)
 One Hot Encoding_linear regression
df_new = pd.get_dummies(df_new[['price','vehicleType','yearOfUsing',
```

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

```
#predict price
#y=ax+b, intercept is b,slope is a
model = LinearRegression().fit(X_train, y_train)
y_pred = regressor.predict(X_test)
errorS = sqrt(mean_squared_error(y_test,y_pred))

print('RMSE:',errorS)
print('R^2:', model.score(X_test, y_test))

#To retrieve the intercept:
print('Intercept:', model.intercept_)

#For retrieving the slope  :

print('Coefficient:',model.coef_)
```

```
RMSE: 4560.420858721138
R^2: 0.6342995799196617
Intercept: 13834.521737543158
Coefficient: [-1.35050750e+02 -2.07370718e+02 -9.41232593e+02  6.14295267e+01
  5.34026364e+00 -7.92053137e-02 -4.81915952e+02  3.86127342e+00
 -1.96725512e+03  5.53021008e-02]
```

```
regressor.predict(X_train)
```

```
array([ 5706.19333666,  -284.08916516, 31027.94579734, ...,
       15419.4043989 , 14020.1718278 ,  1319.46062908])
```

```
X_test_plot= X_test['kilometer']
```

```
mpl.matplotlib_fname()
```

```
'/usr/local/lib/python3.6/dist-packages/matplotlib/mpl-data/matplotlibrc'
```

```
# Visualising the Test set results
plt.figure(figsize=(6,4))
plt.scatter(y_test,y_pred, s = 20, alpha = 0.1)
plt.title('Estimate Price vs.Real Price')
plt.xlabel('Real Price')
plt.ylabel('Predicted Price')

plt.plot([min(y_test),max(y_test)],[min(y_test),max(y_test)],'r')
#plt.plot(X_train, regressor.predict(X_train), color = 'blue')
```

```
plt.tight_layout()
#plt.show()
```

Random Forest

```
#from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
import matplotlib.pyplot as plt

rfr = RandomForestRegressor(n_estimators=10)
rfr.fit(X_train, y_train)
rfr.score(X_test, y_test)
```

```
0.8872268298873156
```

```
y_pred = rfr.predict(X_test)

plt.figure(figsize=(6,4))
plt.scatter(y_test,y_pred,s=20,alpha=0.1)
plt.title('Estimated price vs. real Price')
plt.xlabel('Real Price')
plt.ylabel('Predicted Price')

plt.plot([min(y_test),max(y_test)],[min(y_test),max(y_test)],'r')
plt.tight_layout()
```

Reference

- https://chrisalbon.com/python/data_wrangling/pandas_list_unique_values_in_column/
- https://cmdlinetips.com/2019/10/how-to-drop-rows-based-on-a-column-value-in-pandas-dataframe/
- https://www.analyticsvidhya.com/blog/2018/08/k-nearest-neighbor-introduction-regression-python/
- https://www.youtube.com/watch?v=sA1K22Hmh1g