import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from sklearn import datasets, linear\_model, preprocessing, svm

from sklearn.preprocessing import StandardScaler, Normalizer

import math

import matplotlib

import seaborn as sns

def category\_values(dataframe, categories):

for c in categories:

print('\n', dataframe.groupby(by=c)[c].count().sort\_values(ascending=False))

print('Nulls: ', dataframe[c].isnull().sum())

def plot\_correlation\_map( df ):

corr = df.corr()

\_ , ax = plt.subplots( figsize =( 12 , 10 ) )

cmap = sns.diverging\_palette( 220 , 10 , as\_cmap = True )

\_ = sns.heatmap(

corr,

cmap = cmap,

square=True,

cbar\_kws={ 'shrink' : .9 },

ax=ax,

annot = True,

annot\_kws = { 'fontsize' : 12 }

)

df = pd.read\_csv('autos.csv')

df.describe()

print(df.seller.unique())

print(df.offerType.unique())

print(df.abtest.unique())

print(df.nrOfPictures.unique())

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

print("Too new: %d" % df.loc[df.yearOfRegistration >= 2017].count()['name'])

print("Too old: %d" % df.loc[df.yearOfRegistration < 1950].count()['name'])

print("Too cheap: %d" % df.loc[df.price < 100].count()['name'])

print("Too expensive: " , df.loc[df.price > 150000].count()['name'])

print("Too few km: " , df.loc[df.kilometer < 5000].count()['name'])

print("Too many km: " , df.loc[df.kilometer > 200000].count()['name'])

print("Too few PS: " , df.loc[df.powerPS < 10].count()['name'])

print("Too many PS: " , df.loc[df.powerPS > 500].count()['name'])

print("Fuel types: " , df['fuelType'].unique())

print("Damages: " , df['notRepairedDamage'].unique())

print("Vehicle types: " , df['vehicleType'].unique())

print("Brands: " , df['brand'].unique())

dedups = df.drop\_duplicates(['name','price','vehicleType','yearOfRegistration'

,'gearbox','powerPS','model','kilometer','monthOfRegistration','fuelType'

,'notRepairedDamage'])

dedups = dedups[

(dedups.yearOfRegistration <= 2016)

& (dedups.yearOfRegistration >= 1950)

& (dedups.price >= 100)

& (dedups.price <= 150000)

& (dedups.powerPS >= 10)

& (dedups.powerPS <= 500)]

print("-----------------\nData kept for analisys: %d percent of the entire set\n-----------------" % (100 \* dedups['name'].count() / df['name'].count()))

dedups['notRepairedDamage'].fillna(value='not-declared', inplace=True)

dedups['fuelType'].fillna(value='not-declared', inplace=True)

dedups['gearbox'].fillna(value='not-declared', inplace=True)

dedups['vehicleType'].fillna(value='not-declared', inplace=True)

dedups['model'].fillna(value='not-declared', inplace=True)

dedups.isnull().sum()

categories = ['gearbox', 'model', 'brand', 'vehicleType', 'fuelType', 'notRepairedDamage']

for i, c in enumerate(categories):

v = dedups[c].unique()

g = dedups.groupby(by=c)[c].count().sort\_values(ascending=False)

r = range(min(len(v), 5))

print( g.head())

plt.figure(figsize=(5,3))

plt.bar(r, g.head())

plt.xticks(r, v)

plt.show()

dedups['namelen'] = [min(70, len(n)) for n in dedups['name']]

ax = sns.jointplot(x='namelen',

y='price',

data=dedups[['namelen','price']],

alpha=0.1,

size=8)

labels = ['name', 'gearbox', 'notRepairedDamage', 'model', 'brand', 'fuelType', 'vehicleType']

les = {}

for l in labels:

les[l] = preprocessing.LabelEncoder()

les[l].fit(dedups[l])

tr = les[l].transform(dedups[l])

dedups.loc[:, l + '\_feat'] = pd.Series(tr, index=dedups.index)

labeled = dedups[ ['price'

,'yearOfRegistration'

,'powerPS'

,'kilometer'

,'monthOfRegistration'

, 'namelen']

+ [x+"\_feat" for x in labels]]

len(labeled['name\_feat'].unique()) / len(labeled['name\_feat'])

labeled.drop(['name\_feat'], axis='columns', inplace=True)

plot\_correlation\_map(labeled)

labeled.corr()

labeled.corr().loc[:,'price'].abs().sort\_values(ascending=False)[1:]