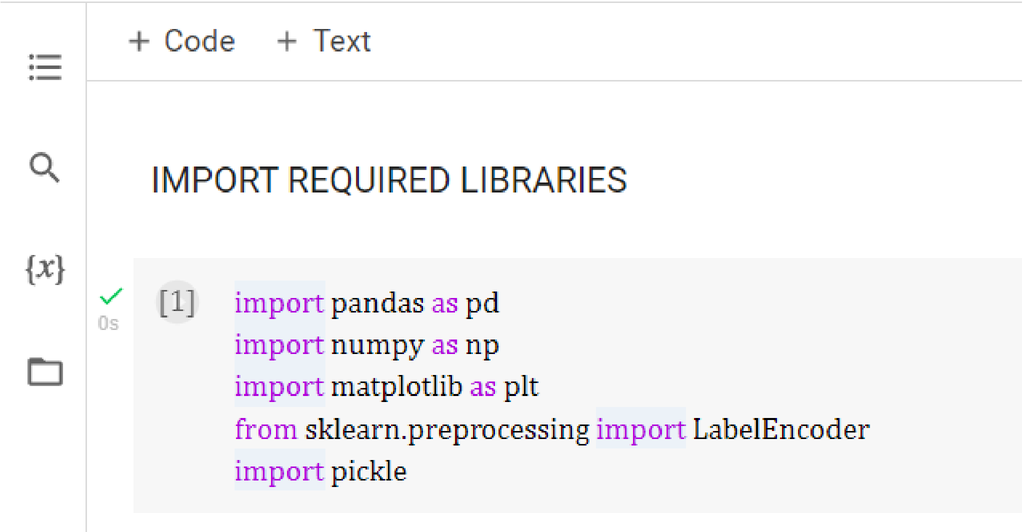
Development Phase
Pre-process the data

Date	12 November 2022
Team ID	PNT2022TMID21553
Project Name	Project – Car Resale Value Prediction

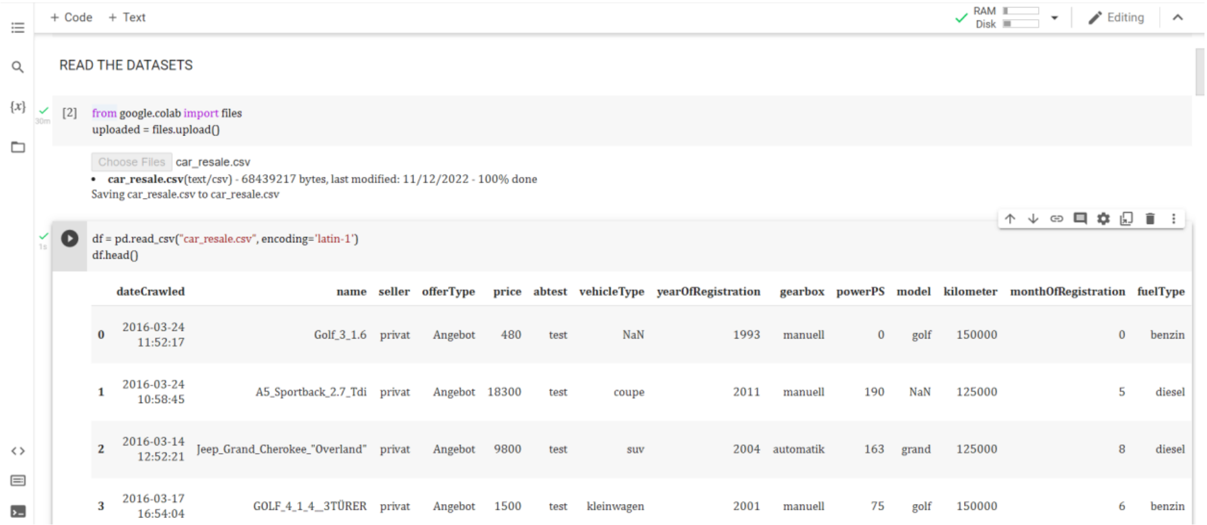
Import the required libraries:



IMPORT REQUIRED LIBRARIES

```
[1] import pandas as pd
    import numpy as np
    import matplotlib as plt
    from sklearn.preprocessing import LabelEncoder
    import pickle
```

Read the dataset:



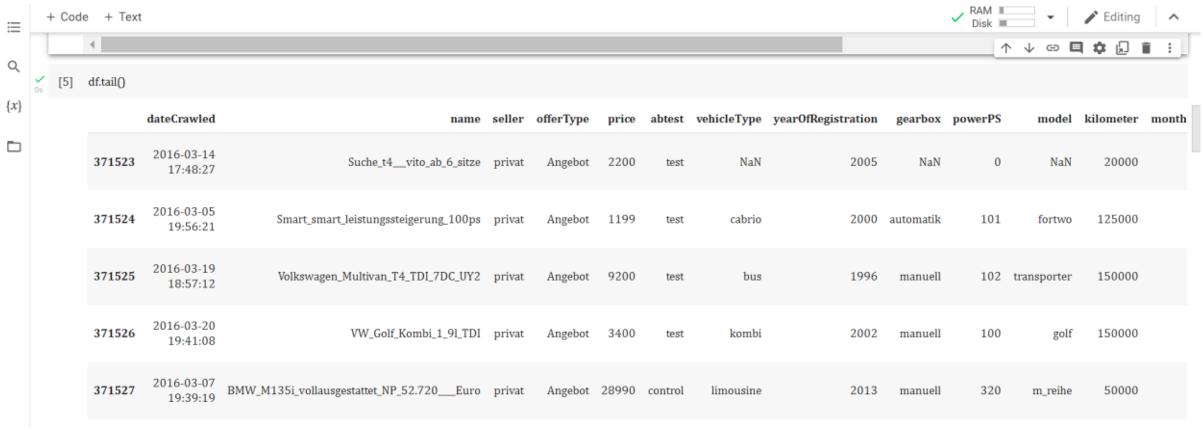
READ THE DATASETS

```
[2] from google.colab import files
    uploaded = files.upload()
```

```
Choose Files car_resale.csv
• car_resale.csv(text/csv) - 68439217 bytes, last modified: 11/12/2022 - 100% done
Saving car_resale.csv to car_resale.csv
```

```
df = pd.read_csv("car_resale.csv", encoding='latin-1')
df.head()
```

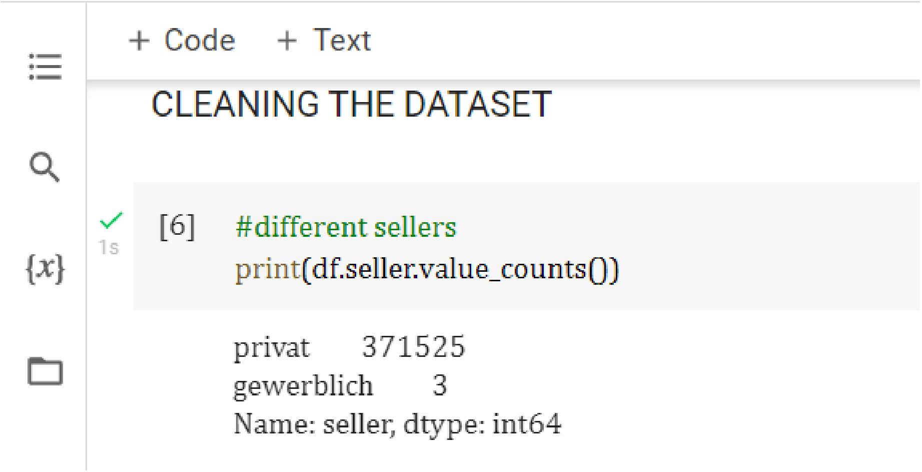
	dateCrawled	name	seller	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthOfRegistration	fuelType
0	2016-03-24 11:52:17	Golf_3_1.6	privat	Angebot	480	test	NaN	1993	manuell	0	golf	150000	0	benzin
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	privat	Angebot	18300	test	coupe	2011	manuell	190	NaN	125000	5	diesel
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	privat	Angebot	9800	test	suv	2004	automatik	163	grand	125000	8	diesel
3	2016-03-17 16:54:04	GOLF_4_1_4__3TÜRER	privat	Angebot	1500	test	kleinwagen	2001	manuell	75	golf	150000	6	benzin



```
[5] df.tail()
```

	dateCrawled	name	seller	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	month
371523	2016-03-14 17:48:27	Suche_t4__vito_ab_6_sitze	privat	Angebot	2200	test	NaN	2005	NaN	0	NaN	20000	
371524	2016-03-05 19:56:21	Smart_smart_leistungssteigerung_100ps	privat	Angebot	1199	test	cabrio	2000	automatik	101	fortwo	125000	
371525	2016-03-19 18:57:12	Volkswagen_Multivan_T4_TDI_7DC_UY2	privat	Angebot	9200	test	bus	1996	manuell	102	transporter	150000	
371526	2016-03-20 19:41:08	VW_Golf_Kombi_1_9l_TDI	privat	Angebot	3400	test	kombi	2002	manuell	100	golf	150000	
371527	2016-03-07 19:39:19	BMW_M135i_vollausgestattet_NP_52.720____Euro	privat	Angebot	28990	control	limousine	2013	manuell	320	m_reihe	50000	

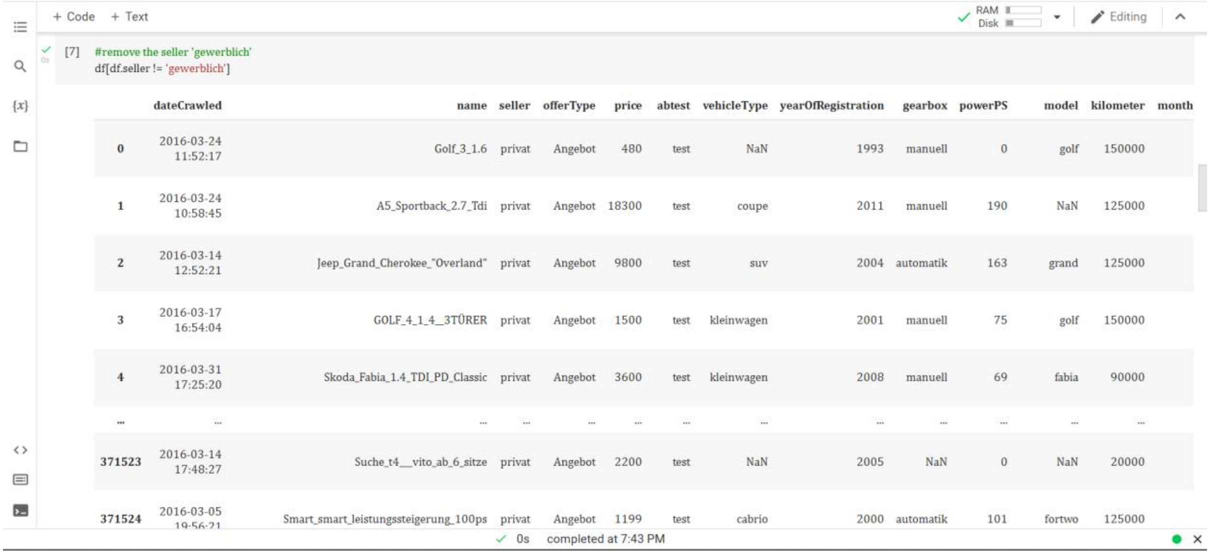
Cleaning the dataset:



CLEANING THE DATASET

```
[6] #different sellers
    print(df.seller.value_counts())
```

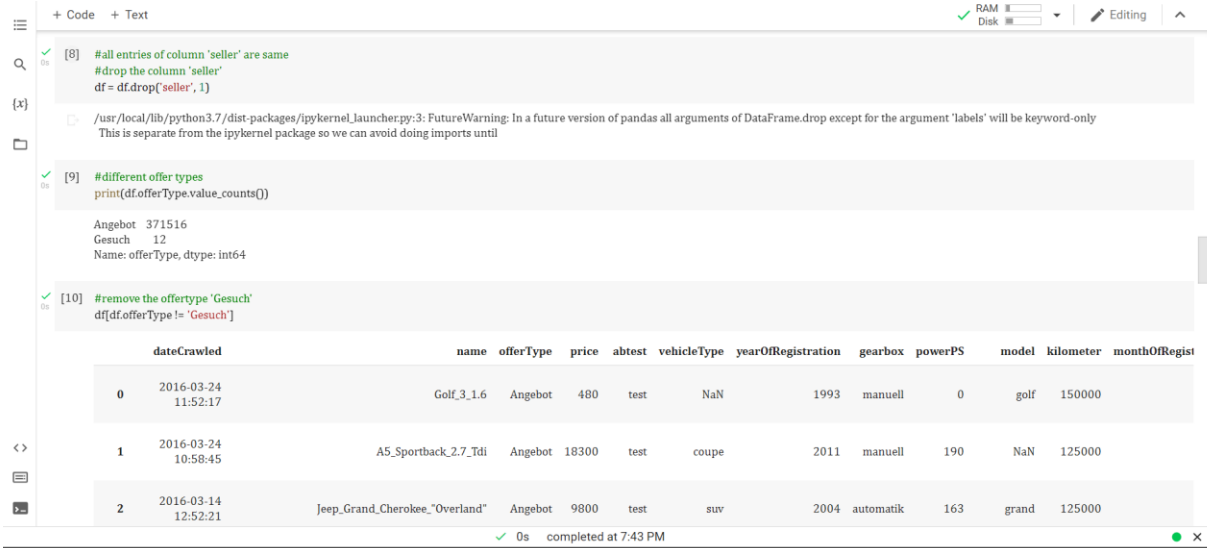
```
privat       371525
gewerblich        3
Name: seller, dtype: int64
```



```
[7] #remove the seller 'gewerblich'
    df[df.seller != 'gewerblich']
```

	dateCrawled	name	seller	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	month
0	2016-03-24 11:52:17	Golf_3_1.6	privat	Angebot	480	test	NaN	1993	manuell	0	golf	150000	
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	privat	Angebot	18300	test	coupe	2011	manuell	190	NaN	125000	
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	privat	Angebot	9800	test	suv	2004	automatik	163	grand	125000	
3	2016-03-17 16:54:04	GOLF_4_1_4__3TÜRER	privat	Angebot	1500	test	kleinwagen	2001	manuell	75	golf	150000	
4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	privat	Angebot	3600	test	kleinwagen	2008	manuell	69	fabia	90000	
...	...	...	...	...	...	...	...	...	...	...	...	...	
371523	2016-03-14 17:48:27	Suche_t4__vito_ab_6_sitze	privat	Angebot	2200	test	NaN	2005	NaN	0	NaN	20000	
371524	2016-03-05 19:56:21	Smart_smart_leistungssteigerung_100ps	privat	Angebot	1199	test	cabrio	2000	automatik	101	fortwo	125000	

✓ 0s completed at 7:43 PM



```
[8]  #all entries of column 'seller' are same
     #drop the column 'seller'
     df = df.drop('seller', 1)
```

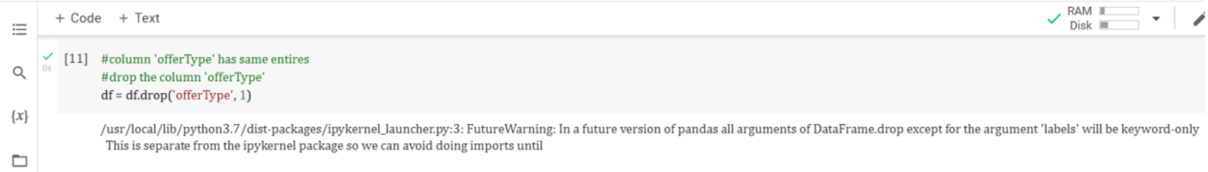
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  This is separate from the ipykernel package so we can avoid doing imports until
```

```
[9]  #different offer types
     print(df.offerType.value_counts())
```

```
Angebot    371516
Gesuch         12
Name: offerType, dtype: int64
```

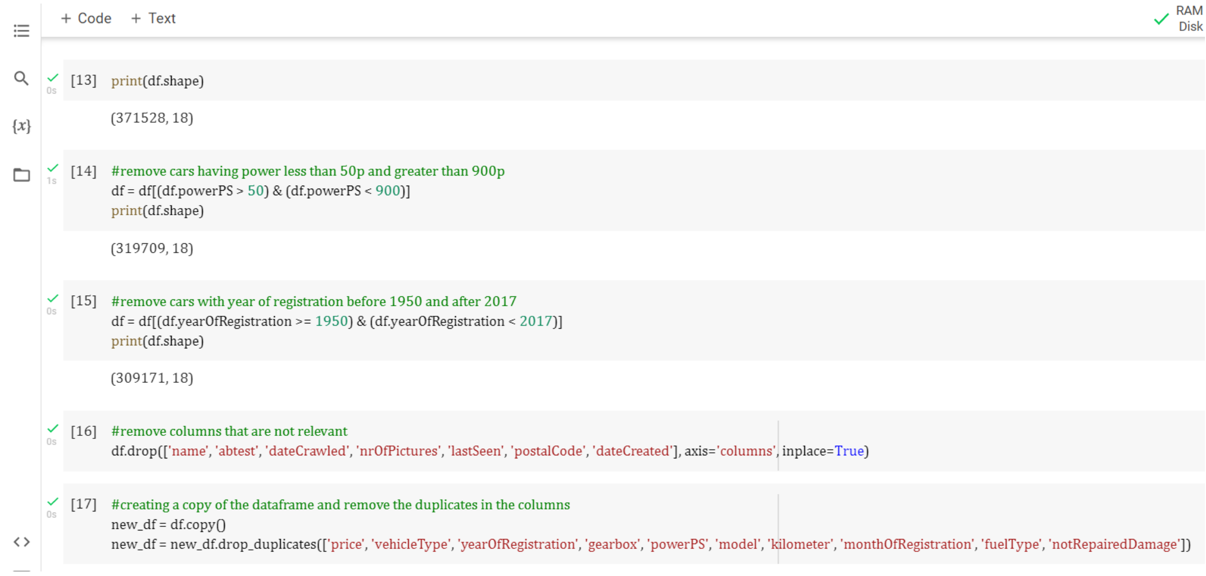
```
[10] #remove the offertype 'Gesuch'
     df[df.offerType != 'Gesuch']
```

	dateCrawled	name	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthOfRegist
0	2016-03-24 11:52:17	Golf_3_1.6	Angebot	480	test	NaN	1993	manuell	0	golf	150000	
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	Angebot	18300	test	coupe	2011	manuell	190	NaN	125000	
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	Angebot	9800	test	suv	2004	automatik	163	grand	125000	



```
[11] #column 'offerType' has same entires
     #drop the column 'offerType'
     df = df.drop('offerType', 1)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: FutureWarning: In a future version of pandas all arguments of DataFrame.drop except for the argument 'labels' will be keyword-only
  This is separate from the ipykernel package so we can avoid doing imports until
```



```
[13] print(df.shape)
```

```
(371528, 18)
```

```
[14] #remove cars having power less than 50p and greater than 900p
     df = df[(df.powerPS > 50) & (df.powerPS < 900)]
     print(df.shape)
```

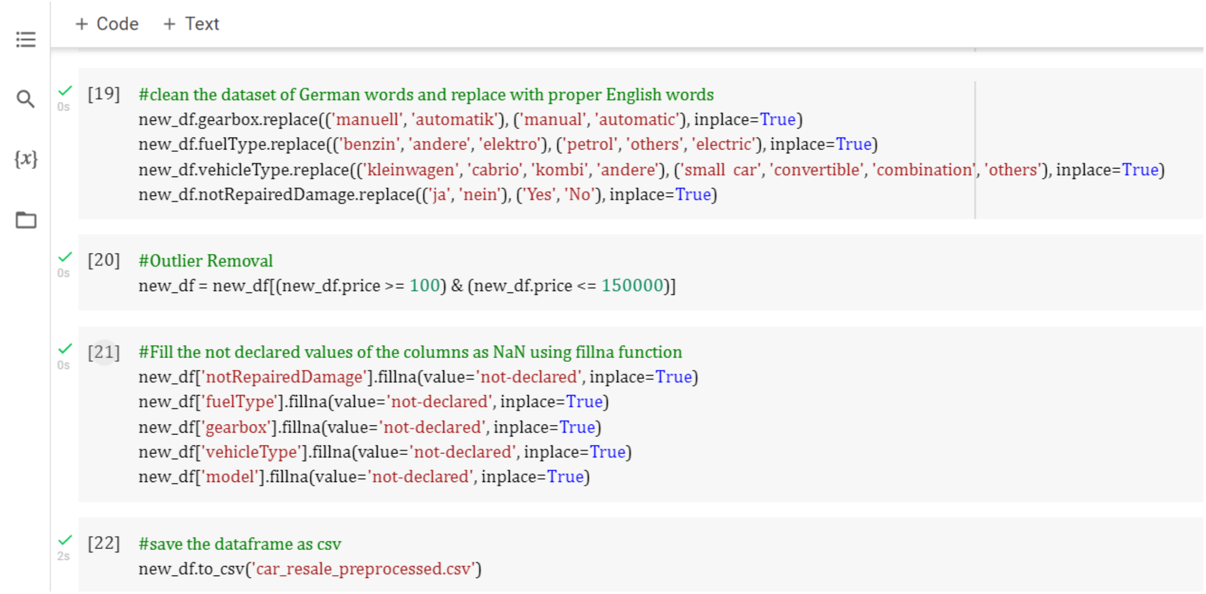
```
(319709, 18)
```

```
[15] #remove cars with year of registration before 1950 and after 2017
     df = df[(df.yearOfRegistration >= 1950) & (df.yearOfRegistration < 2017)]
     print(df.shape)
```

```
(309171, 18)
```

```
[16] #remove columns that are not relevant
     df.drop(['name', 'abtest', 'dateCrawled', 'nrOfPictures', 'lastSeen', 'postalCode', 'dateCreated'], axis='columns', inplace=True)
```

```
[17] #creating a copy of the dataframe and remove the duplicates in the columns
     new_df = df.copy()
     new_df = new_df.drop_duplicates(['price', 'vehicleType', 'yearOfRegistration', 'gearbox', 'powerPS', 'model', 'kilometer', 'monthOfRegistration', 'fuelType', 'notRepairedDamage'])
```



[19]

```
#clean the dataset of German words and replace with proper English words
new_df.gearbox.replace(('manuell', 'automatik'), ('manual', 'automatic'), inplace=True)
new_df.fuelType.replace(('benzin', 'andere', 'elektro'), ('petrol', 'others', 'electric'), inplace=True)
new_df.vehicleType.replace(('kleinwagen', 'cabrio', 'kombi', 'andere'), ('small  car', 'convertible', 'combination', 'others'), inplace=True)
new_df.notRepairedDamage.replace(('ja', 'nein'), ('Yes', 'No'), inplace=True)
```

[20]

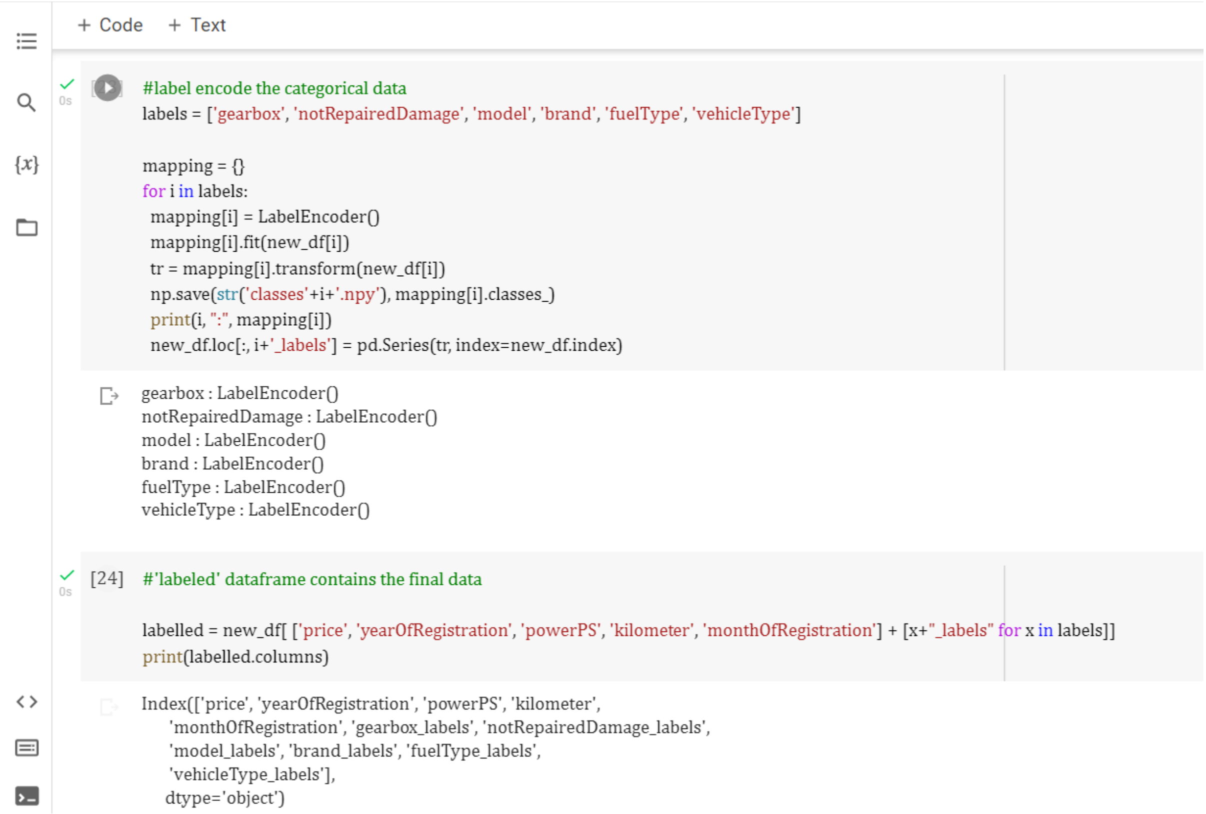
```
#Outlier Removal
new_df = new_df[(new_df.price >= 100) & (new_df.price <= 150000)]
```

[21]

```
#Fill the not declared values of the columns as NaN using fillna function
new_df['notRepairedDamage'].fillna(value='not-declared', inplace=True)
new_df['fuelType'].fillna(value='not-declared', inplace=True)
new_df['gearbox'].fillna(value='not-declared', inplace=True)
new_df['vehicleType'].fillna(value='not-declared', inplace=True)
new_df['model'].fillna(value='not-declared', inplace=True)
```

[22]

```
#save the dataframe as csv
new_df.to_csv('car_resale_preprocessed.csv')
```



```
#label encode the categorical data
labels = ['gearbox', 'notRepairedDamage', 'model', 'brand', 'fuelType', 'vehicleType']

mapping = {}
for i in labels:
  mapping[i] = LabelEncoder()
  mapping[i].fit(new_df[i])
  tr = mapping[i].transform(new_df[i])
  np.save(str('classes'+i+'.npy'), mapping[i].classes_)
  print(i, ":", mapping[i])
  new_df.loc[:, i+'_labels'] = pd.Series(tr, index=new_df.index)
```

```
gearbox : LabelEncoder()
notRepairedDamage : LabelEncoder()
model : LabelEncoder()
brand : LabelEncoder()
fuelType : LabelEncoder()
vehicleType : LabelEncoder()
```

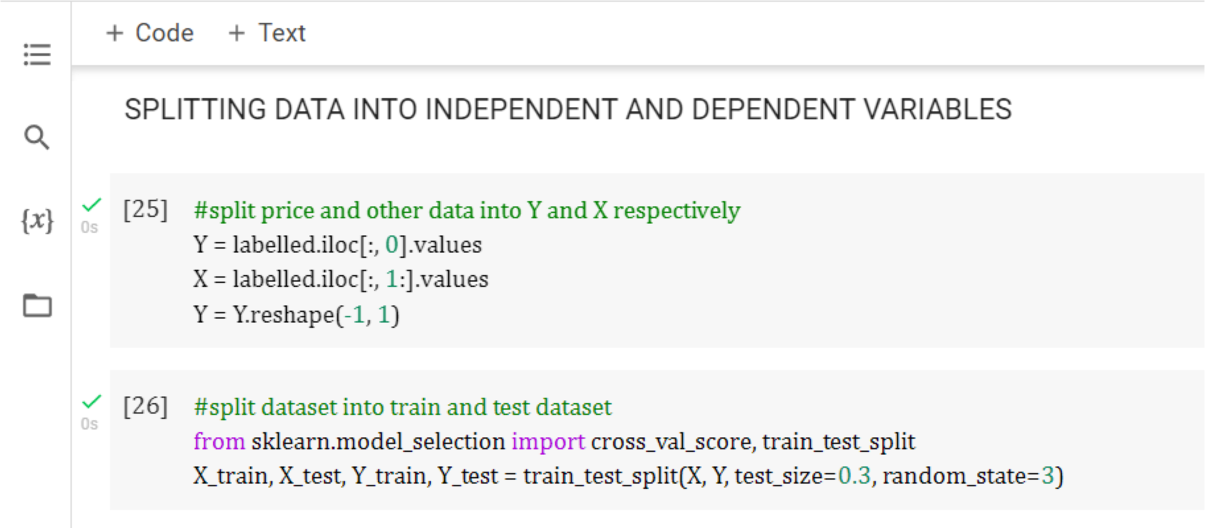
[24]

```
#'labeled' dataframe contains the final data

labelled = new_df[ ['price', 'yearOfRegistration', 'powerPS', 'kilometer', 'monthOfRegistration'] + [x+"_labels" for x in labels]]
print(labelled.columns)
```

```
Index(['price', 'yearOfRegistration', 'powerPS', 'kilometer',
       'monthOfRegistration', 'gearbox_labels', 'notRepairedDamage_labels',
       'model_labels', 'brand_labels', 'fuelType_labels',
       'vehicleType_labels'],
      dtype='object')
```

Splitting the data into independent and dependent variables



SPLITTING DATA INTO INDEPENDENT AND DEPENDENT VARIABLES

```
#split price and other data into Y and X respectively
Y = labelled.iloc[:, 0].values
X = labelled.iloc[:, 1:].values
Y = Y.reshape(-1, 1)
```

```
#split dataset into train and test dataset
from sklearn.model_selection import cross_val_score, train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=3)
```