Development Phase
Sprint 1 delivery

Date	18 November 2022
Team ID	PNT2022TMID21553
Project Name	Project – Car Resale Value Prediction

Sprint 1:

The dataset is collected and is pre-processed.

Once the data is pre-processed, it is split into training and testing dataset and the model is built.

Based on the metrics, out of decision tree classification and random forest regressor, the performance of random forest regressor seemed to be good and thus, random forest regressor model is trained and saved.

Code:

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

```
from google.colab import drive
drive.mount('/content/drive')
df = pd.read_csv("/content/drive/MyDrive/IBM_Nalaiya Thiran/car_resale.csv", encoding='l
atin-1')
df.head()
df.tail()
```

```
#different sellers
print(df.seller.value_counts())
#remove the seller 'gewerblich'
df[df.seller != 'gewerblich']
#all entries of column 'seller' are same
#drop the column 'seller'
df = df.drop('seller', 1)
#different offer types
print(df.offerType.value_counts())
```

```python
#remove the offertype 'Gesuch'
df[df.offerType != 'Gesuch']
#column 'offerType' has same entires
#drop the column 'offerType'
df = df.drop('offerType', 1)
print(df.shape)
#remove cars having power less than 50p and greater than 900p
df = df[(df.powerPS > 50) & (df.powerPS < 900)]
print(df.shape)
#remove cars with year of registration before 1950 and after 2017
df = df[(df.yearOfRegistration >= 1950) & (df.yearOfRegistration < 2017)]
print(df.shape)
#remove columns that are not relevant
df.drop(['name', 'abtest', 'dateCrawled', 'nrOfPictures', 'lastSeen', 'postalCode', 'dateCreated'], axis='columns', inplace=True)
#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'])
#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)
#Outlier Removal
new_df = new_df[(new_df.price >= 100) & (new_df.price <= 150000)]
#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)
#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)
```

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

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

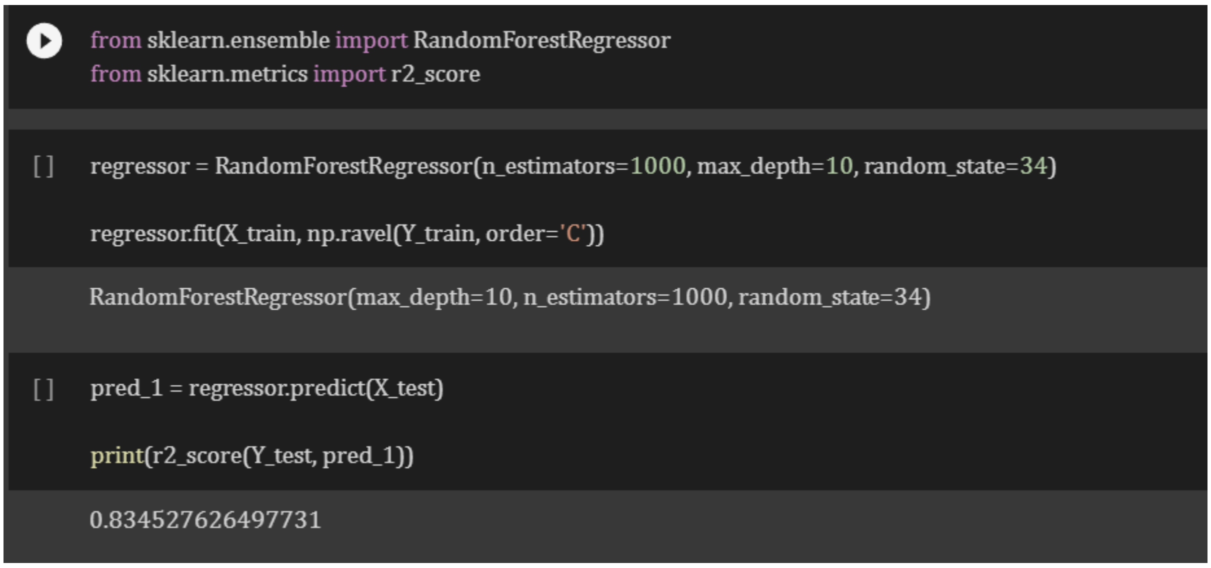
```python
#split price and other data into Y and X respectively
Y = labelled.iloc[:, 0].values
X = labelled.iloc[:, 1:].values
Y = Y.reshape(-1, 1)
#split dataset into train and test dataset
from sklearn.model_selection import cross_val_score, train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3, random_state=3)
```

```python
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
regressor = RandomForestRegressor(n_estimators=1000, max_depth=10, random_state=34)

regressor.fit(X_train, np.ravel(Y_train, order='C'))
pred_1 = regressor.predict(X_test)

print(r2_score(Y_test, pred_1))
```

```python
from sklearn.tree import DecisionTreeClassifier
ds = DecisionTreeClassifier(max_depth=5000, max_features=0.9, max_leaf_nodes=5000, random_state=2, splitter='best')

ds.fit(X_train,  np.ravel(Y_train, order='C'))
pred_3 =ds.predict(X_test)

print(r2_score(Y_test, pred_3))
```

```python
file_name = 'resale_model.pkl'
pickle.dump(regressor, open(file_name, 'wb'))
```

Test case:



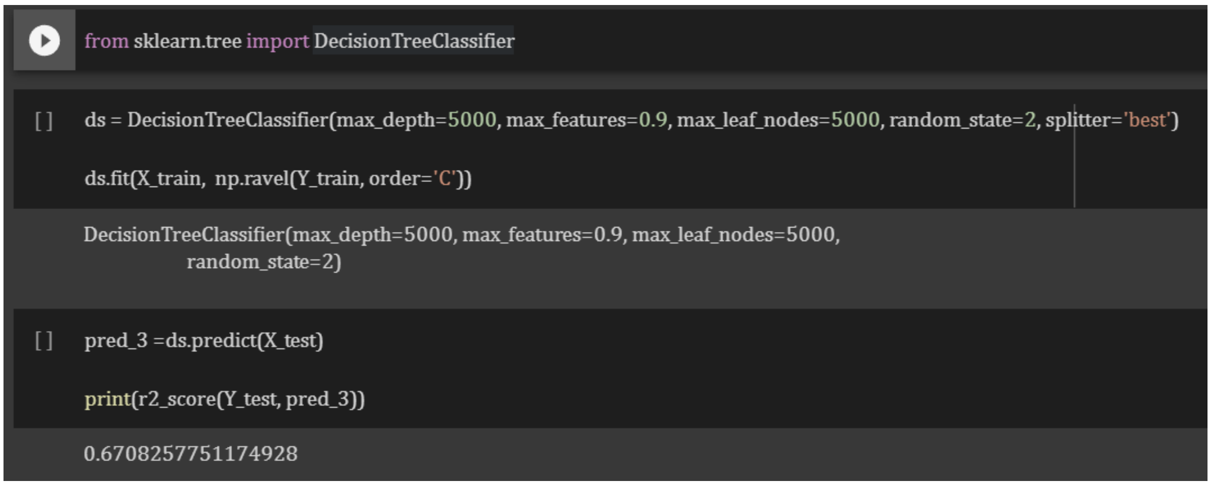
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
```

```
regressor = RandomForestRegressor(n_estimators=1000, max_depth=10, random_state=34)

regressor.fit(X_train, np.ravel(Y_train, order='C'))
```

```
RandomForestRegressor(max_depth=10, n_estimators=1000, random_state=34)
```

```
pred_1 = regressor.predict(X_test)

print(r2_score(Y_test, pred_1))
```

```
0.834527626497731
```



```
from sklearn.tree import DecisionTreeClassifier
```

```
ds = DecisionTreeClassifier(max_depth=5000, max_features=0.9, max_leaf_nodes=5000, random_state=2, splitter='best')

ds.fit(X_train,  np.ravel(Y_train, order='C'))
```

```
DecisionTreeClassifier(max_depth=5000, max_features=0.9, max_leaf_nodes=5000,
                       random_state=2)
```

```
pred_3 =ds.predict(X_test)

print(r2_score(Y_test, pred_3))
```

```
0.6708257751174928
```

The r-squared score of Random Forest regressor is more than the decision tree classifier. So, the regressor model is chosen for prediction.