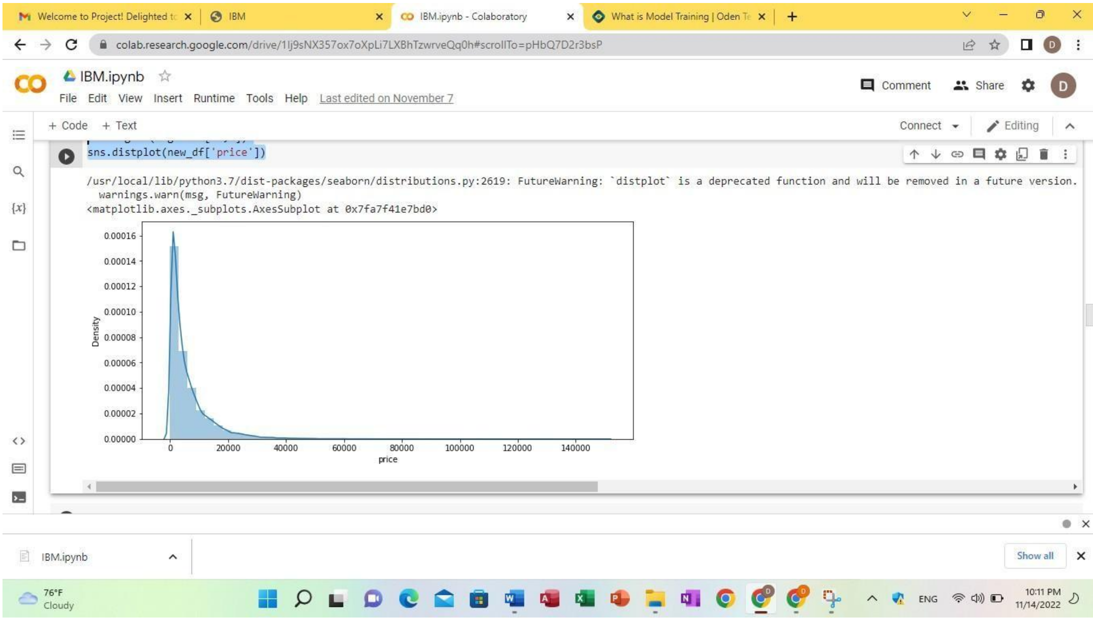
Date	10 November 2022
Team ID	PNT2022TMID01544
Project Name	PROJECT-CAR RESALES VALUE PREDICTION
Maximum Marks	2 Marks

A training model is a dataset that is used to train an algorithm. It consists of the sample output data and the corresponding sets of input data that have an influence on the output. The training model is used to run the input data through the algorithm to correlate the processed output against the sample output. The result from this correlation is used to modify the model.

This iterative process is called “model fitting”. The accuracy of the training dataset or the validation dataset is critical for the precision of the model.

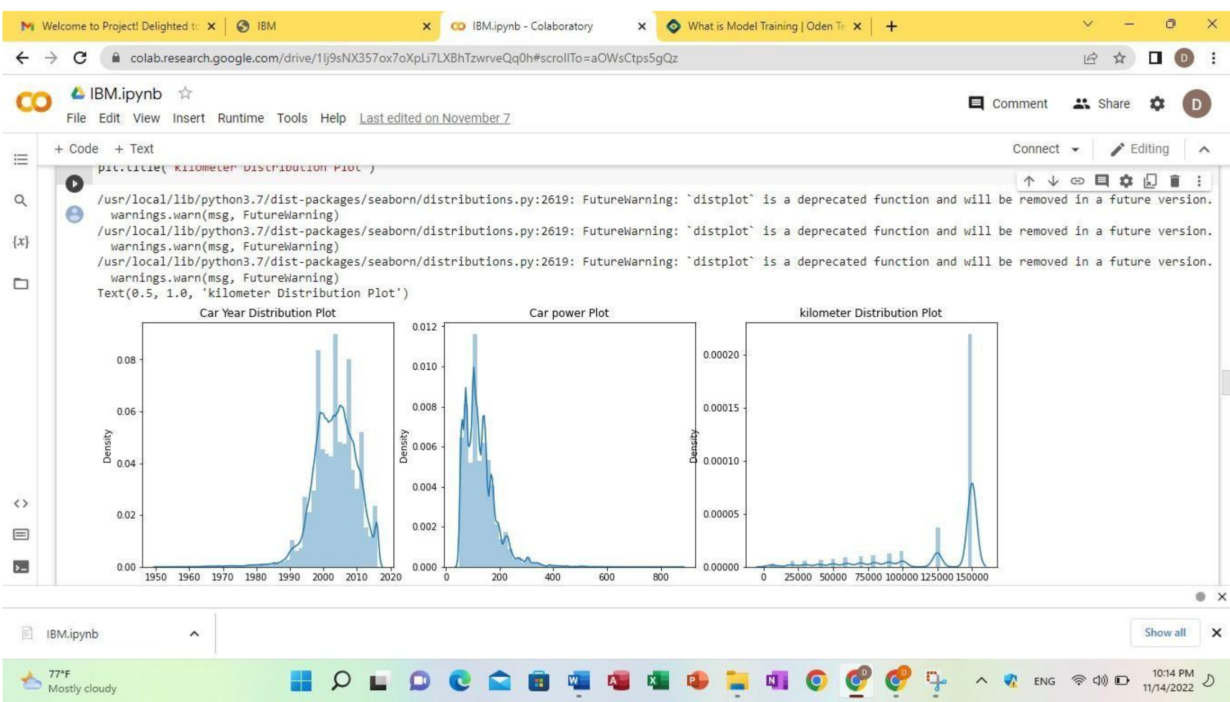
Model training is the process of feeding an algorithm with data to help identify and learn good values for all attributes involved.

```
import seaborn as sns
from matplotlib import *
import sys
from pylab import *
plt.figure(figsize=[11,5])
sns.distplot(new_df['price'])
```

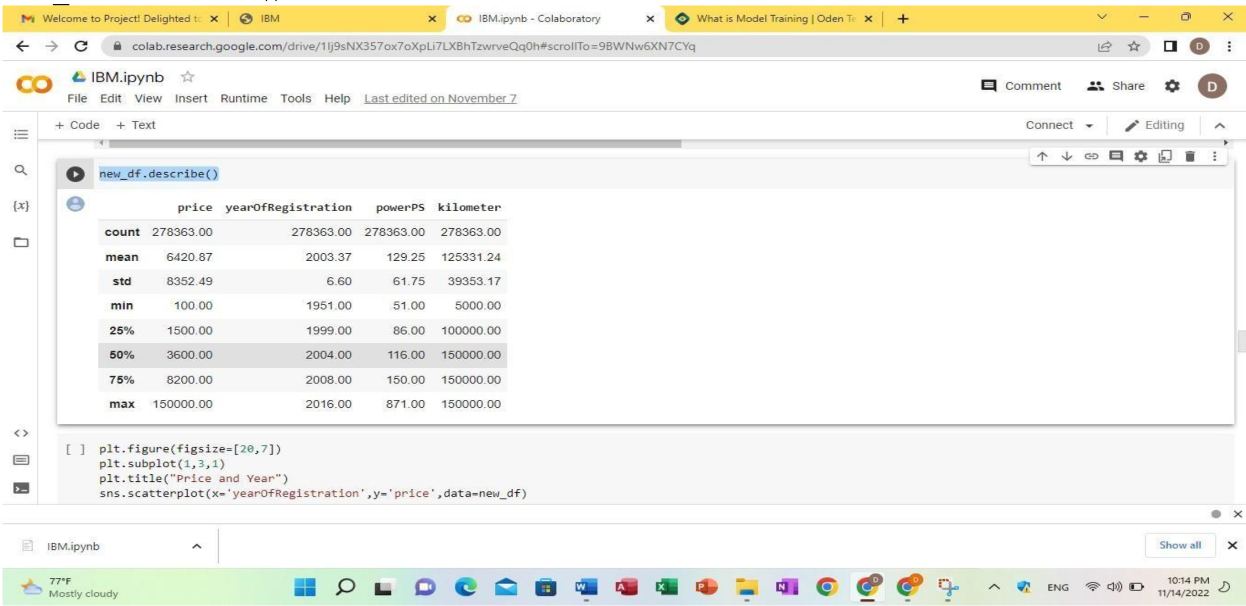


```
plt.figure(figsize=[17,5])
plt.subplot(1,3,1)
sns.distplot(new_df['yearOfRegistration'])
plt.title('Car Year Distribution Plot')

plt.subplot(1,3,2)
sns.distplot(new_df['powerPS'])
plt.title('Car power Plot')
```



```
new_df.describe()
```



```
new_df.describe()
```

	price	yearOfRegistration	powerPS	kilometer
count	278363.00	278363.00	278363.00	278363.00
mean	6420.87	2003.37	129.25	125331.24
std	8352.49	6.60	61.75	39353.17
min	100.00	1951.00	51.00	5000.00
25%	1500.00	1999.00	86.00	100000.00
50%	3600.00	2004.00	116.00	150000.00
75%	8200.00	2008.00	150.00	150000.00
max	150000.00	2016.00	871.00	150000.00

```
plt.figure(figsize=[20,7])
plt.subplot(1,3,1)
plt.title("Price and Year")
sns.scatterplot(x='yearOfRegistration',y='price',data=new_df)
```

```
plt.figure(figsize=[20,7])
plt.subplot(1,3,1)
plt.title("Price and Year")
sns.scatterplot(x='yearOfRegistration',y='price',data=new_df)

plt.subplot(1,3,2)
plt.title("price and kilometer")
sns.scatterplot(x='price',y='kilometer',data=new_df)

plt.subplot(1,3,3)
sns.scatterplot(y='price',x='powerPS',data=new_df)
plt.title("Price and powerPS")
```


```
log_price = np.log(new_df['price'])
new_df['log_price'] = log_price
new_df.head()
```

	seller	offerType	price	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthOfRegistration	fuelType	brand	notRepairedDamage	l
1	privat	Angebot	18300	coupe	2011	manual	190	not-declared	125000	5	diesel	audi	Yes	
2	privat	Angebot	9800	suv	2004	automatic	163	grand	125000	8	diesel	jeep	not-declared	
3	privat	Angebot	1500	small car	2001	manual	75	golf	150000	6	petrol	volkswagen	No	
4	privat	Angebot	3600	small car	2008	manual	69	fabia	90000	7	diesel	skoda	No	
5	privat	Angebot	650	limousine	1995	manual	102	3er	150000	10	petrol	bmw	Yes	

```
plt.figure(figsize=[20,7])
plt.subplot(1,3,1)
plt.title("Price and Year")
sns.scatterplot(x='yearOfRegistration',y='log_price',data=new_df)

plt.subplot(1,3,2)
plt.title("price and kilometer")
sns.scatterplot(x='kilometer',y='log_price',data=new_df)

plt.subplot(1,3,3)
sns.scatterplot(y='price',x='log_price',data=new_df)
plt.title("Price and powerPS")
```

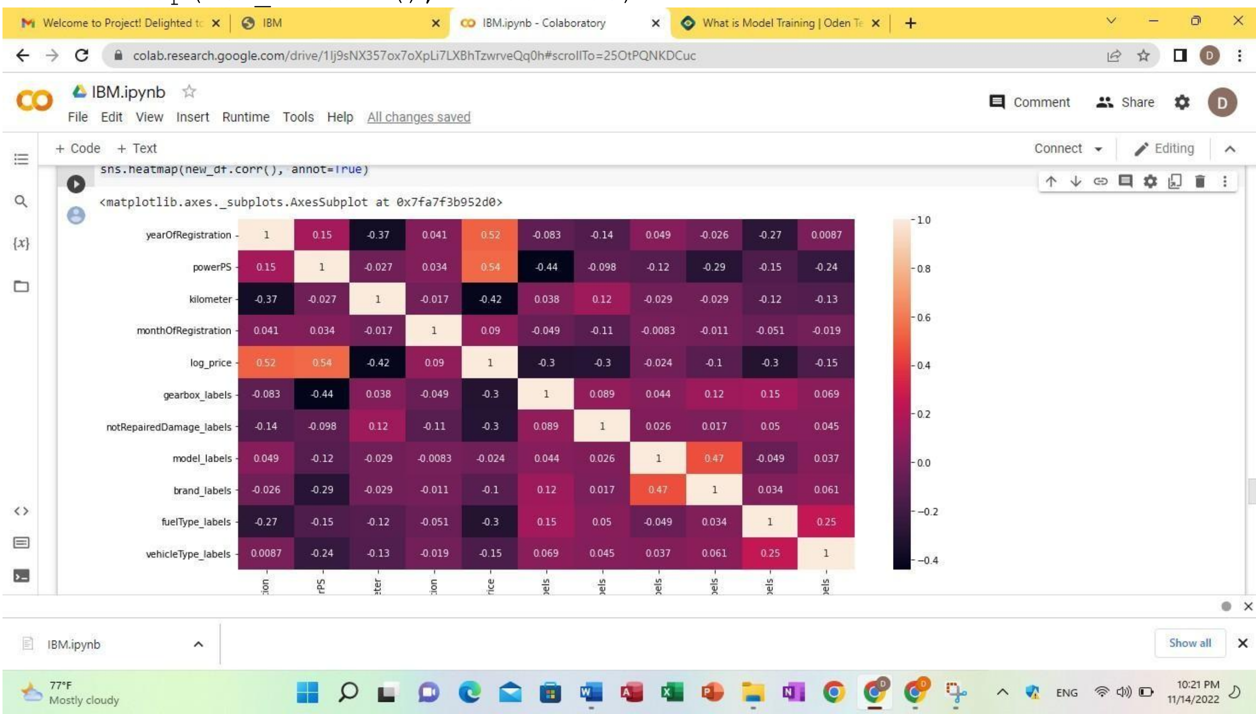

```
new_df= new_df.drop(['price'],axis=1)
new_df['monthOfRegistration']=new_df['monthOfRegistration'].astype(int)
labels= ['gearbox', 'notRepairedDamage', 'model', 'brand', 'fuelType', 've
hicleType']
mapper={}
for i in labels:
  mapper[i] =LabelEncoder()
  mapper[i].fit(new_df[i])
  tr=mapper[i].transform(new_df[i])
  np.save(str('classes'+i+'.npy'), mapper[i].classes_)
  print(i, ":",mapper[i])
  new_df.loc[:, i+'_labels'] = pd.Series (tr, index=new_df.index)

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

```
gearbox : LabelEncoder()
notRepairedDamage : LabelEncoder()
model : LabelEncoder()
brand : LabelEncoder()
fuelType : LabelEncoder()
vehicleType : LabelEncoder()
Index(['log_price', 'yearOfRegistration', 'powerPS', 'kilometer',
       'monthOfRegistration', 'gearbox_labels', 'notRepairedDamage_labels',
       'model_labels', 'brand_labels', 'fuelType_labels',
       'vehicleType_labels'],
      dtype='object')
```

```
plt.figure(figsize=[15,7])
```

```
sns.heatmap(new_df.corr(), annot=True)
```




```
Y  =labeled.iloc[:,0].values
X = labeled.iloc[:,1:].values
Y = Y.reshape(-1,1)
from sklearn.model_selection import train_test_split,cross_val_score
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size=0.3, ran
dom_state=3)

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'))
y_pred=regressor.predict(X_test)
print(r2_score (Y_test,y_pred))
y_pred=regressor.predict(X_test)
print(r2_score (Y_test,y_pred))
df_ev = pd.DataFrame(np.exp(y_pred), columns=['Predicted Price'])

# We can also include the Actual price column in that data frame (so we ca
n manually compare them)
#Y_test=Y_test.reset_index(drop=True)
df_ev['Actual Price'] = np.exp(Y_test)

# we can calculate the difference between the targets and the predictions
```

```
df_ev['Residual'] = df_ev['Actual Price'] - df_ev['Predicted Price']
df_ev['Difference%'] = np.absolute(df_ev['Residual']/df_ev['Actual Price']
*100)

pd.set_option('display.float_format', lambda x: '%.2f' % x)
df_ev.sort_values(by=['Difference%'])

df_ev.tail(5)
```

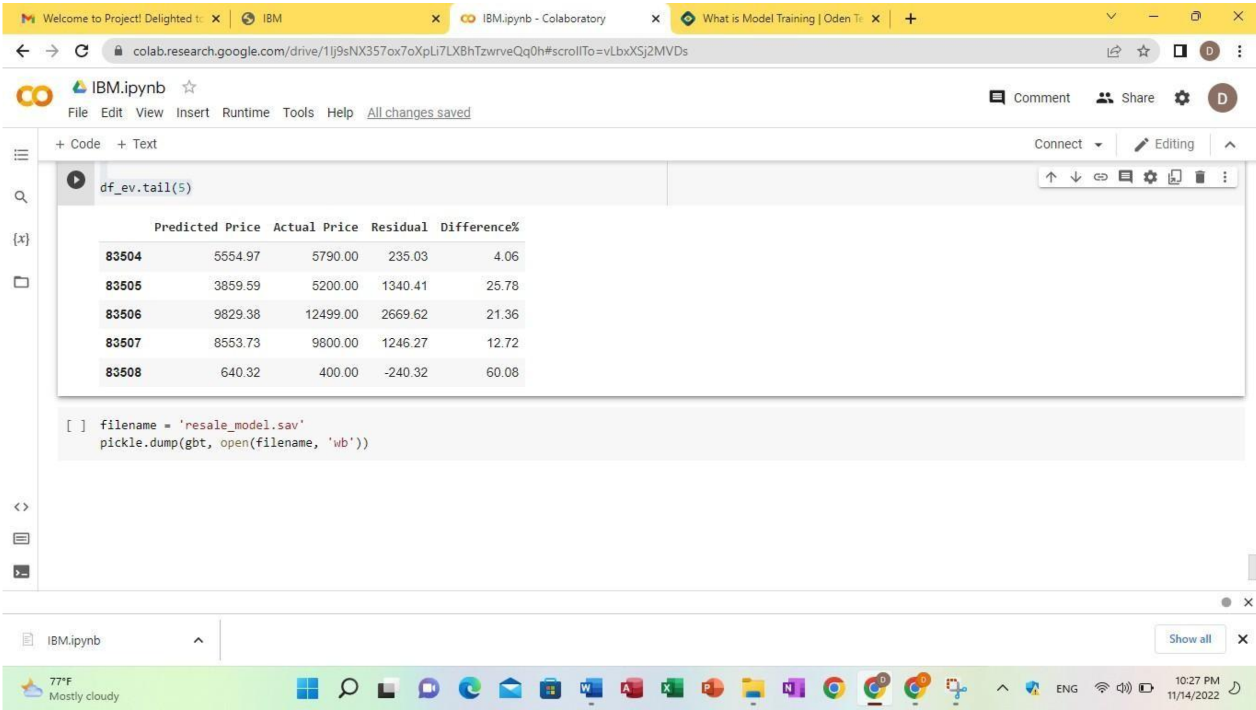
	Predicted Price	Actual Price	Residual	Difference%
83504	4946.32	5790.00	843.68	14.57
83505	4177.92	5200.00	1022.08	19.66
83506	11025.04	12499.00	1473.96	11.79
83507	7967.92	9800.00	1832.08	18.69
83508	564.48	400.00	-164.48	41.12

```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(X_train,Y_train)
y_pred_lr = lr.predict(X_test)
r_squared = r2_score(Y_test,y_pred_lr)
print("R_squared :",r_squared)

from sklearn.ensemble import GradientBoostingRegressor
gbt = GradientBoostingRegressor()
gbt.fit(X_train,Y_train)
y_pred_gbt = gbt.predict(X_test)
r_squared = r2_score(Y_test,y_pred_gbt)
print("R_squared :",r_squared)

df_ev = pd.DataFrame(np.exp(y_pred_gbt), columns=['Predicted Price'])
df_ev['Actual Price'] = np.exp(Y_test)
df_ev['Residual'] = df_ev['Actual Price'] - df_ev['Predicted Price']
df_ev['Difference%'] = np.absolute(df_ev['Residual']/df_ev['Actual Price']
*100)
pd.set_option('display.float_format', lambda x: '%.2f' % x)
df_ev.sort_values(by=['Difference%'])

df_ev.tail(5)
```




```
filename = 'resale_model.sav'
pickle.dump(gbt, open(filename, 'wb'))
```