Mercedes-Benz Greener Manufacturing

Project 1

Patil Deepti Reddy

PG DS – Machine Learning

Problem Description

- Reduce the time a Mercedes-Benz spends on the test bench.
- Problem Statement Scenario:
  Since the first automobile, the Benz Patent Motor Car in 1886, Mercedes-Benz has stood for important automotive innovations. These include the passenger safety cell with a crumple zone, the airbag, and intelligent assistance systems. Mercedes-Benz applies for nearly 2000 patents per year, making the brand the European leader among premium carmakers. Mercedes-Benz is the leader in the premium car industry. With a huge selection of features and options, customers can choose the customized Mercedes-Benz of their dreams.
- To ensure the safety and reliability of every unique car configuration before they hit the road, the company's engineers have developed a robust testing system. As one of the world's biggest manufacturers of premium cars, safety and efficiency are paramount on Mercedes-Benz's production lines. However, optimizing the speed of their testing system for many possible feature combinations is complex and time-consuming without a powerful algorithmic approach.
- You are required to reduce the time that cars spend on the test bench. Others will work with a dataset representing different permutations of features in a Mercedes-Benz car to predict the time it takes to pass testing. Optimal algorithms will contribute to faster testing, resulting in lower carbon dioxide emissions without reducing Mercedes-Benz's standards.
- Following actions should be performed:
- If for any column(s), the variance is equal to zero, then you need to remove those variable(s).
- Check for null and unique values for test and train sets.
- Apply label encoder.
- Perform dimensionality reduction.
- Predict your test_df values using XGBoost.

Import the libraries and read the data

```
import pandas as pd
import numpy as np
# Step1: Read the train.csv and test.csv data
df_test = pd.read_csv('test.csv')
df_train=pd.read_csv("train.csv")
df_train.shape
```

(4209, 378)

```
df_train.describe()
```

	ID	y	X10	X11	X12	X13	X14	X15	X16	X17	...	X375
count	4209.000000	4209.000000	4209.000000	4209.0	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	4209.000000	...	4209.000000
mean	4205.960798	100.669318	0.013305	0.0	0.075077	0.057971	0.428130	0.000475	0.002613	0.007603	...	0.318841
std	2437.608688	12.679381	0.114590	0.0	0.263547	0.233716	0.494867	0.021796	0.051061	0.086872	...	0.466082
min	0.000000	72.110000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
25%	2095.000000	90.820000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
50%	4220.000000	99.150000	0.000000	0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000
75%	6314.000000	109.010000	0.000000	0.0	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	...	1.000000
max	8417.000000	265.320000	1.000000	0.0	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	...	1.000000

8 rows × 370 columns

Preprocess data

#extract useful columns for training by removing ID and Y columns

x_train=df_train.iloc[:,2:]

x_test=df_test.iloc[:,1:]

extract the target column Y

y_train=df_train['y'].values

x_train

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	k	v	at	a	d	u	j	o	0	0	...	0	0	1	0	0	0	0	0	0	0
1	k	t	av	e	d	y	l	o	0	0	...	1	0	0	0	0	0	0	0	0	0
2	az	w	n	c	d	x	j	x	0	0	...	0	0	0	0	0	0	1	0	0	0
3	az	t	n	f	d	x	l	e	0	0	...	0	0	0	0	0	0	0	0	0	0
4	az	v	n	f	d	h	d	n	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	ak	s	as	c	d	aa	d	q	0	0	...	1	0	0	0	0	0	0	0	0	0
4205	j	o	t	d	d	aa	h	h	0	0	...	0	1	0	0	0	0	0	0	0	0
4206	ak	v	r	a	d	aa	g	e	0	0	...	0	0	1	0	0	0	0	0	0	0
4207	al	r	e	f	d	aa	l	u	0	0	...	0	0	0	0	0	0	0	0	0	0
4208	z	r	ae	c	d	aa	g	w	0	0	...	1	0	0	0	0	0	0	0	0	0

4209 rows × 376 columns

Check for null and unique values for test and train sets.

check for any null values

df_train.isna().any()

df_test.isna().any()

```
ID      False
X0      False
X1      False
X2      False
X3      False
        ...
X380    False
X382    False
X383    False
X384    False
X385    False
Length: 377, dtype: bool
```

no null values found in the data.

Check for null and unique values for test and train sets.
If for any column(s), the variance is equal to zero, then you need to remove those variable(s).

```
#Check for unique values for test and train sets and drop them.
for col in x_train.columns:
    car=len(np.unique(x_train[col]))
    if car==1:
        print(col)
        x_train.drop(col, axis=1,inplace=True)
        x_test.drop(col, axis=1, inplace=True)
```

Output

```
X11
X93
X107
X233
X235
X268
X289
X290
X293
X297
X330
X347
```

The output shows the columns having only one value. As, the variance is equal to zero, these columns are removed from train and test dataset.

Apply label encoder.

```
# Import label encoder
from sklearn import preprocessing
# label_encoder object knows how to understand word labels.
label_encoder = preprocessing.LabelEncoder()

for col in x_train.columns:
    typ = df_train[col].dtype
    if typ == 'object':
        # Encode labels in column 'species'.
        x_train[col]= label_encoder.fit_transform(x_train[col])
        x_test[col]= label_encoder.fit_transform(x_test[col])
```

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X12	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	32	23	17	0	3	24	9	14	0	0	...	0	0	1	0	0	0	0	0	0	0
1	32	21	19	4	3	28	11	14	0	0	...	1	0	0	0	0	0	0	0	0	0
2	20	24	34	2	3	27	9	23	0	0	...	0	0	0	0	0	0	1	0	0	0
3	20	21	34	5	3	27	11	4	0	0	...	0	0	0	0	0	0	0	0	0	0
4	20	23	34	5	3	12	3	13	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4204	8	20	16	2	3	0	3	16	0	0	...	1	0	0	0	0	0	0	0	0	0
4205	31	16	40	3	3	0	7	7	0	0	...	0	1	0	0	0	0	0	0	0	0
4206	8	23	38	0	3	0	6	4	0	1	...	0	0	1	0	0	0	0	0	0	0
4207	9	19	25	5	3	0	11	20	0	0	...	0	0	0	0	0	0	0	0	0	0
4208	46	19	3	2	3	0	6	22	0	0	...	1	0	0	0	0	0	0	0	0	0

4209 rows × 364 columns

- The columns which are not of type integers are converted into integer values using labelEncoder.

Perform dimensionality reduction.

```
#Perform dimensionality reduction.
from sklearn.decomposition import PCA

sklearn_pca = PCA(n_components=0.95, random_state=420)
sklearn_pca.fit(x_train)
x_train_transformed=sklearn_pca.transform(x_train)
print(x_train_transformed.shape)
 (4209, 6)
x_test_transformed=sklearn_pca.transform(x_test)
print(x_test_transformed.shape)
```

- The above code is used to reduce the dimensions of train and test data from 364 features to 6 features.

Predict your test_df values using XGBoost

```
from xgboost import XGBRegressor
from sklearn import model_selection
model = XGBRegressor(objective='reg:squarederror', n_estimators=1000)
model.fit(x_train_transformed, y_train)
y_pred=model.predict(x_train_transformed)
from sklearn.metrics import r2_score, mean_squared_error
from math import sqrt
print(sqrt(mean_squared_error(y_train, y_pred)))
```

2.14438850331057

```
y_test_pred=model.predict(x_test_transformed)
```

- The test_df values predicted are-

array([81.000244, 97.20449 , 84.81835 , ..., 103.20449 , 108.65345 , 102.384254], dtype=float32)

Conclusion

- The model is trained using Mercedes-Benz data using XGBoost to predict the testing time.
- Before training, the data was preprocessed to find missing and unique values, and the dimensionality reduction was done.