LOVELY PROFESSIONAL UNIVERSITY

Machine Learning CA 1 Report:

On

Mercedes Benz Greener Manufacturing

Name: Gudladhana Harshith

APRIL 2022

School of Computer Science & Engineering

Submitted to- Dr. Rahul

Subject Code: INT247

Name: Gudladhana Harshith

Reg.no:11903248

Roll.no: RKM045A03

Table of Content:

Abstract:

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:

(i) If for any column(s), the variance is equal to zero, then you need to remove those variable(s).
(ii) Check for null and unique values for test and train sets.
(iii) Apply label encoder.
(iv) Perform dimensionality reduction.
(v) Predict your test_df values using XGBoost.

Introduction:

Since the first automobile, the Benz Patent Motor Car in 1886, Mercedes-Benz has stood for important automotive innovations. These include, for example, the passenger safety cell with 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. Daimler's Mercedes-Benz cars are leaders 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 each unique car configuration before they hit the road, Daimler's engineers have developed a robust testing system. But, optimizing the speed of their testing system for so many possible feature combinations are complex and time-consuming without a powerful algorithmic approach. As one of the world's biggest manufacturers of premium cars, safety and efficiency are paramount on Daimler's production lines.

We must tackle the curse of dimensionality and reduce the time that cars spend on the test bench. Competitors will work with a dataset representing different permutations of Mercedes-Benz car features to predict the time it takes to pass testing. Winning algorithms will contribute to speedier testing, resulting in lower carbon dioxide emissions without reducing Daimler's standards.

File Description:

Variables with letters are categorical. Variables with 0/1 are binary values.

1. **train.csv** — the training set

2. **test.csv** — the test set, you must predict the 'y' variable for the 'ID's in this file

3. **sample_submission.csv** — a sample submission file in the correct format

Machine Learning Problem Formulation:

This dataset contains an anonymized set of variables, each representing a custom feature in a Mercedes car. For example, a variable could be 4WD, added air suspension, or a head-up display.

The ground truth is labelled ‘y’ and represents the time (in seconds) that the car took to pass testing for each variable.

As ‘y’ is a continuous variable, hence we can formulate the problem as a regression problem.

The performance metric used here is:

R2 (Coefficient of determination): It is the proportion of the variance in the dependent variable that is predictable from the independent variables.

Tools and Libraries used:

These are frameworks in python to handle commonly required tasks.

Pandas: For handling structured data Scikit Learn: For machine learning

NumPy: For linear algebra and mathematics

Seaborn: For data visualization and exploratory data analysis Matplotlib: is a graphical plotting library for python

XGBoost: provides a high-performance implementation of gradient boosted decision tree

Approach Used: As ‘y’ is a continuous variable, so it is a regression problem. Initially, I can think of the Linear Regression approach as my benchmark model because it is the simplest regression algorithm we have.

So, our initial approach is as follow:

1. Importing essential libraries

2. Perform EDA for understanding dataset and feature analysis

3. We must perform dimensionality reduction techniques as Mercedes.So,We can get a better understanding by analysing the data.

4. Standardization and featurization of train data

5. Preparing test data

6. Train the baseline model

7. Evaluate the result based on R2 Score

8. Based on the result of the baseline model, try more complex models for further improvement.

Screenshots:

1) Importing the required libraries to work up on the data.

```
In [2]: # Importing the required libraries
        import numpy as np
        import pandas as pd
        import seaborn as sns
        import matplotlib.pyplot as plt
        from sklearn.decomposition import PCA
```

2)observe the starting 5 values of the data of both train and test datasets.

In [3]:
```
train.head()
```

Out[3]:

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	0	130.81	k	v	at	a	d	u	j	o	...	0	0	1	0	0	0	0	0	0	0
1	6	88.53	k	t	av	e	d	y	l	o	...	1	0	0	0	0	0	0	0	0	0
2	7	76.26	az	w	n	c	d	x	j	x	...	0	0	0	0	0	0	1	0	0	0
3	9	80.62	az	t	n	f	d	x	l	e	...	0	0	0	0	0	0	0	0	0	0
4	13	78.02	az	v	n	f	d	h	d	n	...	0	0	0	0	0	0	0	0	0	0

5 rows × 378 columns

In [4]:
```
test.head()
```

Out[4]:

	ID	X0	X1	X2	X3	X4	X5	X6	X8	X10	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	1	az	v	n	f	d	t	a	w	0	...	0	0	0	1	0	0	0	0	0	0
1	2	t	b	ai	a	d	b	g	y	0	...	0	0	1	0	0	0	0	0	0	0
2	3	az	v	as	f	d	a	j	j	0	...	0	0	0	1	0	0	0	0	0	0
3	4	az	l	n	f	d	z	l	n	0	...	0	0	0	1	0	0	0	0	0	0
4	5	w	s	as	c	d	y	i	m	0	...	1	0	0	0	0	0	0	0	0	0

5 rows × 377 columns

3)observe the statistics of the train and test datasets.

In [4]:
```
train.describe()
test.describe()
```

4) i)now observe and print the size of the training and testing datasets.

ii)collect the Y values(Variable values) in to array and print that array.

iii)observe the type of data in the datasets

In [5]:
```
print('Size of training set: {} rows and {} columns'.format(*train.shape))
print('Size of testing set: {} rows and {} columns'.format(*test.shape))
```

```
Size of training set: 4209 rows and 378 columns
Size of testing set: 4209 rows and 377 columns
```

In [6]:
```
# Collect the Y values into an array
y_train = train['y'].values
```

In [7]:
```
y_train
```

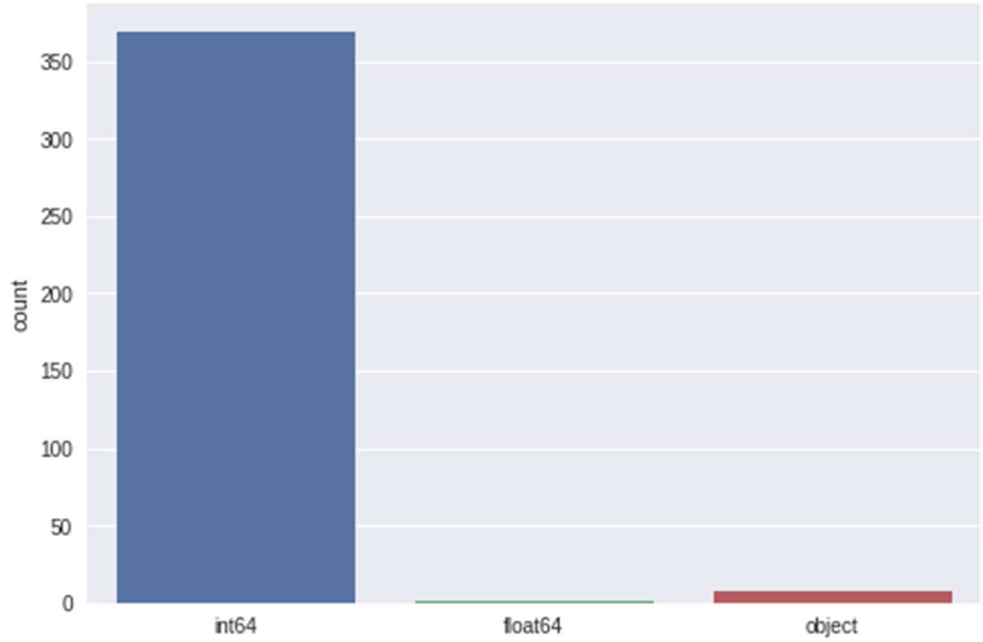
Out[7]:
```
array([130.81,  88.53,  76.26, ..., 109.22,  87.48, 110.85])
```

In [8]:
```
# Understand the data types
cols = [c for c in train.columns if 'X' in c]
print('Number of features: {}'.format(len(cols)))
print('Feature types:')
train[cols].dtypes.value_counts()
```

Out[8]:
```
Number of features: 376
Feature types:
int64     368
object      8
dtype: int64
```

In [4]:
```
sns.countplot(train.dtypes)
```

With the help of the given code, we can observe the type of the data statistical.



We have plotted the type of the data in module present in the seaborn as Count plot, which count the frequency of the data types and plot it.

In [9]:
```
# Count the data in each of the columns

counts = [[], [], []]
for c in cols:
    typ = train[c].dtype
    uniq = len(np.unique(train[c]))
    if uniq == 1:
        counts[0].append(c)
    elif uniq == 2 and typ == np.int64:
        counts[1].append(c)
    else:
        counts[2].append(c)
print('Constant features: {} Binary features: {} Categorical features: {}\n'
 .format(*[len(c) for c in counts]))
print('Constant features:', counts[0])
print('Categorical features:', counts[2])
```

```
Constant features: 12 Binary features: 356 Categorical features: 8

Constant features: ['X11', 'X93', 'X107', 'X233', 'X235', 'X268', 'X289', 'X290', 'X293', 'X297', 'X330', 'X347']
Categorical features: ['X0', 'X1', 'X2', 'X3', 'X4', 'X5', 'X6', 'X8']
```

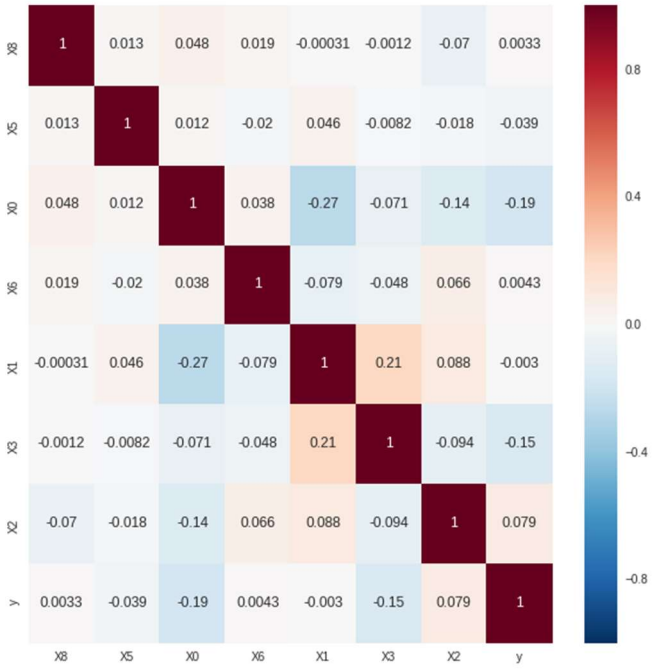
We will count the data in each column by iterating through it.

If we get constant features, we will append then into list1

If we get categorical features, we will append then into list2

```
plt.figure(figsize=(10,10))
top_features=['X8','X5','X0','X6','X1','X3','X2','y']
train_top_features_df=train[top_features]
test_top_features_df=test[top_features[0:len(top_features)-1]]
sns.heatmap(train_top_features_df.corr(), annot=True)
```

We will plot a heatmap to find the relationship among the variables**.** low-value show in low-intensity colour and high-value show in hight-intensity colour format.



In [10]:
```
# Splitting the data
usable_columns = list(set(train.columns) - set(['ID', 'y']))
y_train = train['y'].values
id_test = test['ID'].values
x_train = train[usable_columns]
x_test = test[usable_columns]
```

Check for null values and unique values for train & test data

In [11]:
```
def check_missing_values(df):
    if df.isnull().any().any():
        print('There are missing values in the dataframe')
    else:
        print('There are no missing values in the dataframe')
```

In [12]:
```
check_missing_values(x_train)
check_missing_values(x_test)
```

```
There are no missing values in the dataframe
There are no missing values in the dataframe
```

Label Encoding the categorical values

In [13]:
```
for column in usable_columns:
    cardinality = len(np.unique(x_train[column]))
    if cardinality == 1:
        x_train.drop(column, axis=1) # Column with only one
        # value is useless so we drop it
        x_test.drop(column, axis=1)
    if cardinality > 2: # Column is categorical
        mapper = lambda x: sum([ord(digit) for digit in x])
        x_train[column] = x_train[column].apply(mapper)
        x_test[column] = x_test[column].apply(mapper)
x_train.head()
```

We will split the data into two parts 1) training data 2) testing data

We will check for the null values if we found any null values, we will print them out.

We will add label encoding for the categorical data.

In [17]:
```
x_train,x_val,y_train,y_val = train_test_split(pca2_results_train, y_train, test_size=0.2, random_state=4242)
```

In [18]:
```
d_train = xgb.DMatrix(x_train,label = y_train)
d_val = xgb.DMatrix(x_val,label = y_val)

# dtest = xgb.DMatrix(x_test)

d_test = xgb.DMatrix(pca2_results_test)
```

We will apply the train test split for the data. We will split the data and apply the XG boost algorithm.

In [19]:
```
params = {}
params['objective'] = 'reg:linear'
params['eta'] = 0.02
params['max_depth'] = 4

def xgb_r2_score(preds, dtrain):
    labels = dtrain.get_label()
    return 'r2', r2_score(labels, preds)
watchlist = [(d_train, 'train'), (d_val, 'valid')]
clf = xgb.train(params, d_train,  1000, watchlist, early_stopping_rounds=50,
 feval=xgb_r2_score, maximize=True, verbose_eval=10)
```

We will add the prediction to another csv file.

Predict test_df using XGBoost

In [20]:
```
p_test = clf.predict(d_test)
```

In [21]:
```
sub = pd.DataFrame()
sub['ID'] = id_test
sub['y'] = p_test
sub.to_csv('test_df.csv', index = False)
sub.head()
```

Out[21]:

	ID	y
0	1	82.844788
1	2	97.869873
2	3	82.781380
3	4	77.284958
4	5	113.026222

References:

1. https://ieeexplore.ieee.org/document/7506650
2. https://www.appliedaicourse.com/
3. https://www.kaggle.com/c/mercedes-benz-greener-manufacturing/discussion
4. https://www.kaggle.com/c/mercedes-benz-greener-manufacturing/discussion/36390
5. https://medium.com/analytics-vidhya/mercedes-benz-greener-manufacturing-kaggle-competition-1c25c89e012

GitHub Link:

Project GitHub Link