In [ ]:
```
import numpy as np
import pandas as pd
from sklearn.decomposition import PCA
import xgboost as xgb
from sklearn.metrics import r2_score
from sklearn.model_selection import train_test_split
```

In [ ]:
```
df_train = pd.read_csv(r"Z:\simp\ML\Mercedes-Benz_Greener_Manufacturing-master\train.csv")
df_train.head()
df_test = pd.read_csv(r"Z:\simp\ML\Mercedes-Benz_Greener_Manufacturing-master\test.csv")
df_test.head()
```

In [ ]:
```
y_train = df_train['y'].values
usable_columns = list(set(df_train.columns) - set(['ID', 'y']))
id_test = df_test['ID'].values
x_train = df_train[usable_columns]
x_test = df_test[usable_columns]
```

In [ ]:
```
# If for any column(s), the variance is equal to zero, then you need to remove those variable(s).

# Apply label encoder.

for column in usable_columns:
    cardinality = len(np.unique(x_train[column]))
    if cardinality == 1:
        x_train.drop(column, axis=1)
        x_test.drop(column, axis=1)
    if cardinality > 2:
        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()
```

In [ ]:
```
# Check for null and unique values for test and train sets.

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")
check_missing_values(df_train)
check_missing_values(df_test)
```

In [ ]:
```
# Perform dimensionality reduction.

pca = PCA(n_components=12, random_state=420)
pca2_results_train = pca.fit_transform(x_train)
pca2_results_test = pca.transform(x_test)
```

In [ ]:
```
# Training using xgboost

x_train, x_valid, y_train, y_valid = train_test_split(pca2_results_train,y_train,test_size=0.2,random_state=4242
```

In [ ]:
```
d_train = xgb.DMatrix(x_train, label=y_train)
d_valid = xgb.DMatrix(x_valid, label=y_valid)
d_test = xgb.DMatrix(pca2_results_test)
```

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

In [ ]:
```
def xgb_r2_score(preds, dtrain):
    labels = dtrain.get_label()
    return 'r2', r2_score(labels, preds)
```

In [ ]:
```
watchlist = [(d_train, 'train'), (d_valid, 'valid')]
```

In [ ]:
```
clf = xgb.train(params, d_train,
                1000, watchlist, early_stopping_rounds=50,
                feval=xgb_r2_score, maximize=True, verbose_eval=10)
```

In [ ]:
```
# Predict your test_df values using XGBoost.

p_test = clf.predict(d_test)
```

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

In [ ]:
```
sub.head()
```