```
#Beacon Nicolas Mkhabele Project 1 - Mercedes-Benz Greener Manufacturing
#Project 1 - Mercedes-Benz Greener Manufacturing


from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import numpy as np
import pandas as pd
import tensorflow as tf
import matplotlib.pyplot as plt
%matplotlib inline
from patsy import dmatrices
import sklearn
import seaborn as sns
from scipy.stats import skew, norm
from scipy.stats import chi2_contingency
from sklearn.preprocessing import LabelEncoder
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import xgboost as xgb
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
from sklearn import model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression



train_data = pd.read_csv('train.csv')
test_data = pd.read_csv('test.csv')


train_data.head()
```

	ID	y	X0	X1	X2	X3	X4	X5	X6	X8	...	X375	X376	X377	X378	X379	X3
0	0	130.81	32	23	17	0	3	24	9	14	...	0	0	1	0	0	
1	6	88.53	32	21	19	4	3	28	11	14	...	1	0	0	0	0	
2	7	76.26	20	24	34	2	3	27	9	23	...	0	0	0	0	0	
3	9	80.62	20	21	34	5	3	27	11	4	...	0	0	0	0	0	
4	13	78.02	20	23	34	5	3	12	3	13	...	0	0	0	0	0	

5 rows × 366 columns

```
test_data.head()
```

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

5 rows × 377 columns

```
#If for any column(s), the variance is equal to zero, then you need to remove those variable(s).
train_variances = train_data.var()
print(train_variances)
```

```
ID      5.941936e+06
y       1.607667e+02
X0      1.887419e+02
X1      7.277797e+01
X2      1.188081e+02
            ...
X380    8.014579e-03
```

```
X382    7.546747e-03
X383    1.660732e-03
X384    4.750593e-04
X385    1.423823e-03
Length: 366, dtype: float64
```

```
test_variances = test_data.var()

print(test_variances)
```

```
ID      5.871311e+06
X10     1.865006e-02
X11     2.375861e-04
X12     6.885074e-02
X13     5.734498e-02
           ...
X380    8.014579e-03
X382    8.715481e-03
X383    4.750593e-04
X384    7.124196e-04
X385    1.660732e-03
Length: 369, dtype: float64
<ipython-input-23-86e55a16b82a>:1: FutureWarning: The default value of numeric_only in DataFrame.var is deprecated. In a future version
  test_variances = test_data.var()
```

```
zero_var_cols_train = train_variances[train_variances == 0].index
print(zero_var_cols_train)
```

```
Index([], dtype='object')
```

```
zero_var_cols_test = test_variances[test_variances == 0].index
print(zero_var_cols_test)
```

```
Index(['X257', 'X258', 'X295', 'X296', 'X369'], dtype='object')
```

```
train_data = train_data.drop(columns=zero_var_cols_train)
print(train_data)
```

```
        ID       y  X0 X1  X2 X3 X4  X5 X6 X8  ...  X375  X376  X377  X378  \
0        0  130.81   k  v  at  a  d   u  j  o  ...     0     0     1     0
1        6   88.53   k  t  av  e  d   y  l  o  ...     1     0     0     0
2        7   76.26  az  w   n  c  d   x  j  x  ...     0     0     0     0
3        9   80.62  az  t   n  f  d   x  l  e  ...     0     0     0     0
4       13   78.02  az  v   n  f  d   h  d  n  ...     0     0     0     0
...    ...     ...  .. ..  .. .. ..  .. .. ..  ...   ...   ...   ...   ...
4204  8405  107.39  ak  s  as  c  d  aa  d  q  ...     1     0     0     0
4205  8406  108.77   j  o   t  d  d  aa  h  h  ...     0     1     0     0
4206  8412  109.22  ak  v   r  a  d  aa  g  e  ...     0     0     1     0
4207  8415   87.48  al  r   e  f  d  aa  l  u  ...     0     0     0     0
4208  8417  110.85   z  r  ae  c  d  aa  g  w  ...     1     0     0     0

      X379  X380  X382  X383  X384  X385
0        0     0     0     0     0     0
1        0     0     0     0     0     0
2        0     0     1     0     0     0
3        0     0     0     0     0     0
4        0     0     0     0     0     0
...    ...   ...   ...   ...   ...   ...
4204     0     0     0     0     0     0
4205     0     0     0     0     0     0
4206     0     0     0     0     0     0
4207     0     0     0     0     0     0
4208     0     0     0     0     0     0

[4209 rows x 366 columns]
```

```
zero_var_cols_test = test_variances[test_variances == 0].index
print(zero_var_cols_test)
```

```
Index(['X257', 'X258', 'X295', 'X296', 'X369'], dtype='object')
```

```
# Check for null and unique values for test and train sets
# Checking for null values in the train set
train_null_counts = train_data.isnull().sum()
```

```
print("Null values in the train set:")
print(train_null_counts)
```

```
Null values in the train set:
ID      0
y       0
X0      0
X1      0
X2      0
       ..
X380    0
X382    0
X383    0
X384    0
X385    0
Length: 366, dtype: int64
```

```
# Checking for null values in the test set
test_null_counts = test_data.isnull().sum()
print("Null values in the test set:")
print(test_null_counts)
```

```
Null values in the test set:
ID      0
X0      0
X1      0
X2      0
X3      0
       ..
X380    0
X382    0
X383    0
X384    0
X385    0
Length: 377, dtype: int64
```

```
# Checking for unique values in the train set
train_unique_counts = train_data.nunique()
print("Unique values in the train set:")
print(train_unique_counts)
```

```
Unique values in the train set:
ID      4209
y       2545
X0        47
X1        27
X2        44
         ...
X380       2
X382       2
X383       2
X384       2
X385       2
Length: 366, dtype: int64
```

```
# CheckING for unique values in the test set
test_unique_counts = test_data.nunique()
print("Unique values in the test set:")
print(test_unique_counts)
```

```
Unique values in the test set:
ID      4209
X0        49
X1        27
X2        45
X3         7
         ...
X380       2
X382       2
X383       2
X384       2
X385       2
Length: 377, dtype: int64
```

```
# CreatING a LabelEncoder object by understanding
label_encoder = LabelEncoder()
```

```
# Applying label encoder to categorical columns in the train dataset
for column in train_data.columns:
    if train_data[column].dtype == 'object':
        train_data[column] = label_encoder.fit_transform(train_data[column])


# Applying label encoder to categorical columns in the test dataset
for column in test_data.columns:
    if test_data[column].dtype == "object":
        test_data[column] = label_encoder.fit_transform(test_data[column])




# Separatingn target variable from features in the train dataset
target = train_data['ID']
train_features = train_data.drop('ID', axis=1)
print(target)
```

```
     0          0
     1          6
     2          7
     3          9
     4         13
             ...
     4204    8405
     4205    8406
     4206    8412
     4207    8415
     4208    8417
     Name: ID, Length: 4209, dtype: int64
```

```
print(train_features)
```

```
                y  X0  X1  X2  X3  X4  X5  X6  X8  X10  ...  X375  X376  X377  \
     0     130.81  32  23  17   0   3  24   9  14    0  ...     0     0     1
     1      88.53  32  21  19   4   3  28  11  14    0  ...     1     0     0
     2      76.26  20  24  34   2   3  27   9  23    0  ...     0     0     0
     3      80.62  20  21  34   5   3  27  11   4    0  ...     0     0     0
     4      78.02  20  23  34   5   3  12   3  13    0  ...     0     0     0
     ...      ...  ..  ..  ..  ..  ..  ..  ..  ..  ...  ...   ...   ...   ...
     4204  107.39   8  20  16   2   3   0   3  16    0  ...     1     0     0
     4205  108.77  31  16  40   3   3   0   7   7    0  ...     0     1     0
     4206  109.22   8  23  38   0   3   0   6   4    0  ...     0     0     1
     4207   87.48   9  19  25   5   3   0  11  20    0  ...     0     0     0
     4208  110.85  46  19   3   2   3   0   6  22    0  ...     1     0     0

           X378  X379  X380  X382  X383  X384  X385
     0        0     0     0     0     0     0     0
     1        0     0     0     0     0     0     0
     2        0     0     0     1     0     0     0
     3        0     0     0     0     0     0     0
     4        0     0     0     0     0     0     0
     ...    ...   ...   ...   ...   ...   ...   ...
     4204     0     0     0     0     0     0     0
     4205     0     0     0     0     0     0     0
     4206     0     0     0     0     0     0     0
     4207     0     0     0     0     0     0     0
     4208     0     0     0     0     0     0     0

     [4209 rows x 365 columns]
```

```
# Combining train and test datasets
combined_data = pd.concat([train_features, test_data])
print(combined_data)
```

```
                y  X0  X1  X2  X3  X4  X5  X6  X8  X10  ...  X107  X233  X235  \
     0     130.81  32  23  17   0   3  24   9  14    0  ...   NaN   NaN   NaN
     1      88.53  32  21  19   4   3  28  11  14    0  ...   NaN   NaN   NaN
     2      76.26  20  24  34   2   3  27   9  23    0  ...   NaN   NaN   NaN
     3      80.62  20  21  34   5   3  27  11   4    0  ...   NaN   NaN   NaN
     4      78.02  20  23  34   5   3  12   3  13    0  ...   NaN   NaN   NaN
     ...      ...  ..  ..  ..  ..  ..  ..  ..  ..  ...  ...   ...   ...   ...
     4204     NaN   6   9  17   5   3   1   9   4    0  ...   0.0   0.0   0.0
     4205     NaN  42   1   8   3   3   1   9  24    0  ...   0.0   0.0   0.0
     4206     NaN  47  23  17   5   3   1   3  22    0  ...   0.0   0.0   0.0
     4207     NaN   7  23  17   0   3   1   2  16    0  ...   0.0   0.0   0.0
     4208     NaN  42   1   8   2   3   1   6  17    0  ...   0.0   0.0   0.0

           X268  X289  X290  X293  X297  X330  X347
     0      NaN   NaN   NaN   NaN   NaN   NaN   NaN
```

```
1      NaN   NaN   NaN   NaN   NaN   NaN   NaN
2      NaN   NaN   NaN   NaN   NaN   NaN   NaN
3      NaN   NaN   NaN   NaN   NaN   NaN   NaN
4      NaN   NaN   NaN   NaN   NaN   NaN   NaN
...    ...   ...   ...   ...   ...   ...   ...
4204   0.0   0.0   0.0   0.0   0.0   0.0   0.0
4205   0.0   0.0   0.0   0.0   0.0   0.0   0.0
4206   0.0   0.0   0.0   0.0   0.0   0.0   0.0
4207   0.0   0.0   0.0   0.0   0.0   0.0   0.0
4208   0.0   0.0   0.0   0.0   0.0   0.0   0.0

[8418 rows x 378 columns]
```

```
# Performing feature scaling
scaler = StandardScaler()
scaled_data = scaler.fit_transform(combined_data)
print(scaled_data)
```

```
[[ 2.37742381  0.12839426  1.39462106 ...         nan         nan
           nan]
 [-0.95751994  0.12839426  1.16034042 ...         nan         nan
           nan]
 [-1.92534776 -0.69908276  1.51176138 ...         nan         nan
           nan]
 ...
 [        nan  1.16274054  1.39462106 ... -0.01541566 -0.01541566
  -0.02180363]
 [        nan -1.5955162   1.39462106 ... -0.01541566 -0.01541566
  -0.02180363]
 [        nan  0.81795845 -1.18246599 ... -0.01541566 -0.01541566
  -0.02180363]]
```

```
X_train = train_data.drop('ID', axis=1)
y_train = train_data['ID']
```

```
pca = PCA(n_components=2)
X_train_reduced = pca.fit_transform(X_train)
X_test_reduced = pca.fit_transform(test_data)

print(X_train_reduced)
```

```
[[ 14.64981315  24.16061024]
 [ -5.81324995 -12.39023806]
 [  2.08732912 -29.77327191]
 ...
 [ 29.66515988  -5.63712965]
 [ 13.85407413 -21.98582379]
 [-10.42795398  16.12260911]]
```

```
print(X_test_reduced)
```

```
[[ 4209.99886501    14.89369   ]
 [ 4209.04419469   -14.80267873]
 [ 4208.05774157    12.35667214]
 ...
 [-4201.91992115   -13.75688807]
 [-4202.92240483    24.62146632]
 [-4204.91338235   -15.67147299]]
```

```
#Perform dimensionality reduction.
# Printing the shape of the reduced datasets
print('Train data shape after dimensionality reduction:', X_train_reduced.shape)
```

```
Train data shape after dimensionality reduction: (4209, 2)
```

```
print('Test data shape after dimensionality reduction:', X_test_reduced.shape)
```

```
Test data shape after dimensionality reduction: (4209, 2)
```

```
# Initialize the XGBoost classifier
model = xgb.XGBRegressor()
```

```
# Fit the model on the train dataset
model.fit(X_train,y_train)
```

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, gpu_id=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=None, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=None, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             n_estimators=100, n_jobs=None, num_parallel_tree=None,
             predictor=None, random_state=None, ...)
```

```
#Predict your test_df values using xgboost

model=LinearRegression().fit(X_train,y_train)

X_pred = model.predict(X_train)


# Print the predicted values
print(X_pred)
```

```
[3.4375000e-01 3.2617500e+03 1.1303750e+03 ... 2.6097500e+03 3.7431875e+03
 3.0474375e+03]
```