Document

Of

Mercedes-Benz Greener Manufacturing .

Project 1

DESCRIPTION - :

Reduce the time a Mercedes-Benz spends on the test bench.

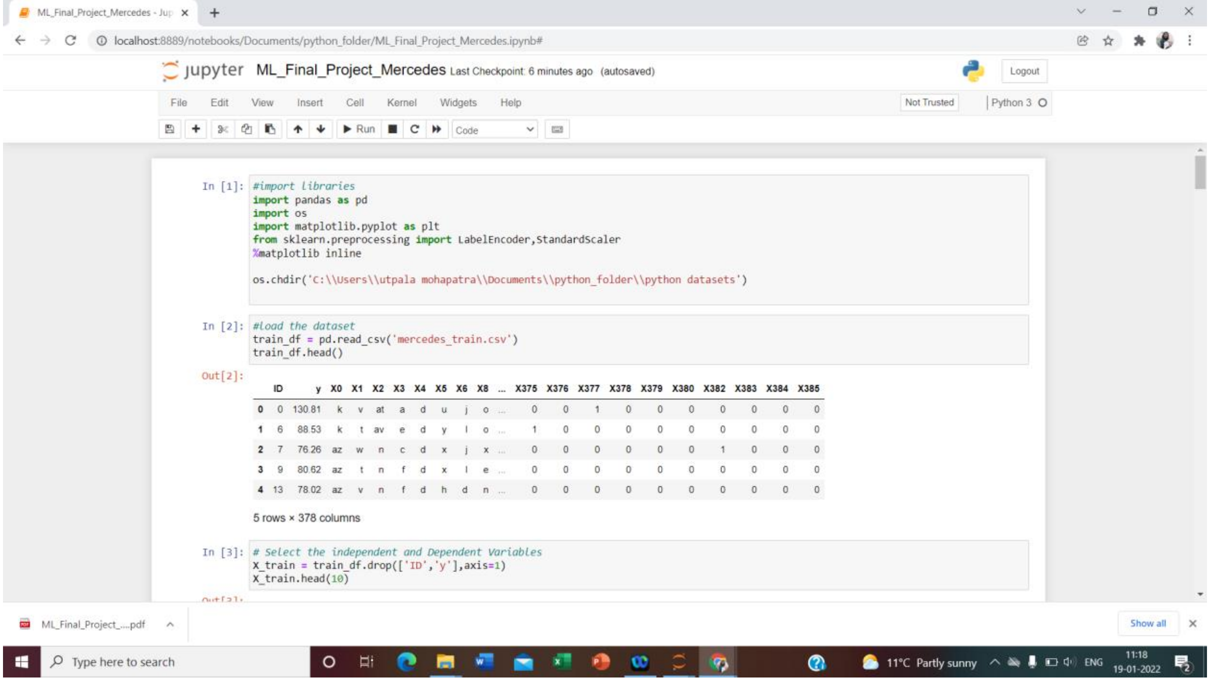
Problem Statement Scenario:
Since the first automobile, the Benz Patent Motor Car in 1886, Mercedes-Benz has stood for important automotive innovations. These include the passenger safety cell with a crumple zone, the airbag, and intelligent assistance systems. Mercedes-Benz applies for nearly 2000 patents 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.

Tasks - :

1- **Load train dataset**



In [1]:
```
#import libraries
import pandas as pd
import os
import matplotlib.pyplot as plt
from sklearn.preprocessing import LabelEncoder,StandardScaler
%matplotlib inline

os.chdir('C:\\Users\\utpala mohapatra\\Documents\\python_folder\\python datasets')
```

In [2]:
```
#load the dataset
train_df = pd.read_csv('mercedes_train.csv')
train_df.head()
```

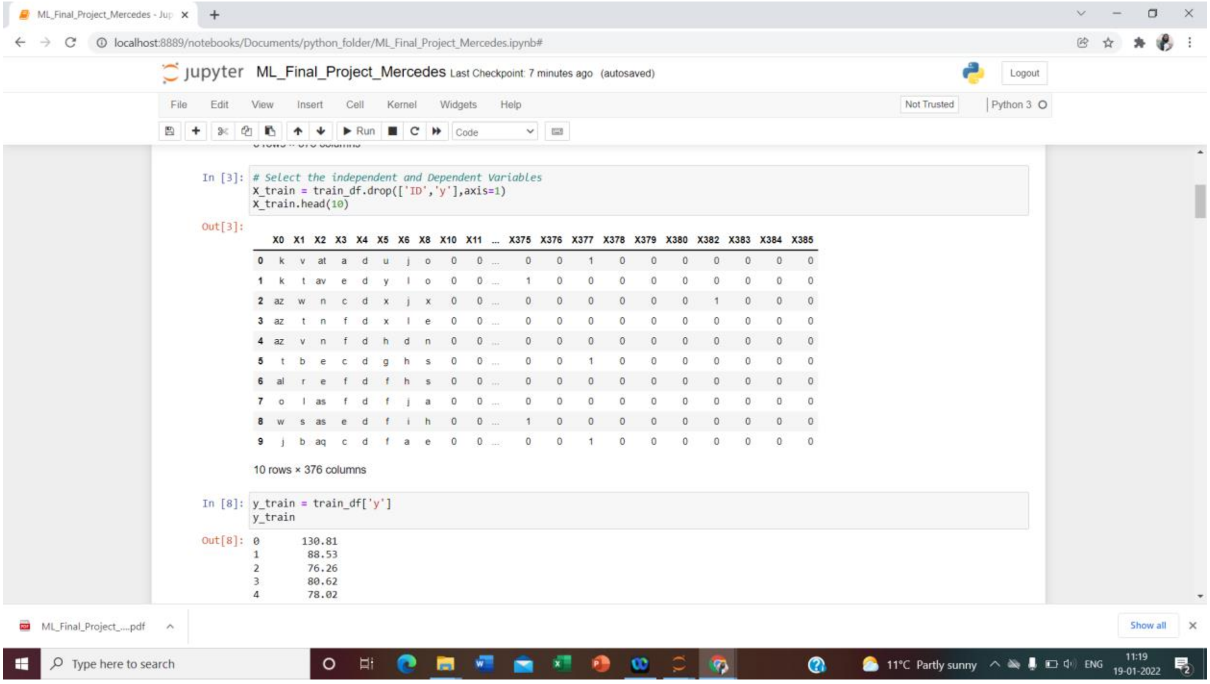
Out[2]:

	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 [3]:
```
# Select the independent and Dependent Variables
X_train = train_df.drop(['ID','y'],axis=1)
X_train.head(10)
```

2- Select Dependent and independent variables



jupyter ML_Final_Project_Mercedes Last Checkpoint: 7 minutes ago (autosaved)

In [3]:
```
# Select the independent and Dependent Variables
X_train = train_df.drop(['ID','y'],axis=1)
X_train.head(10)
```

Out[3]:

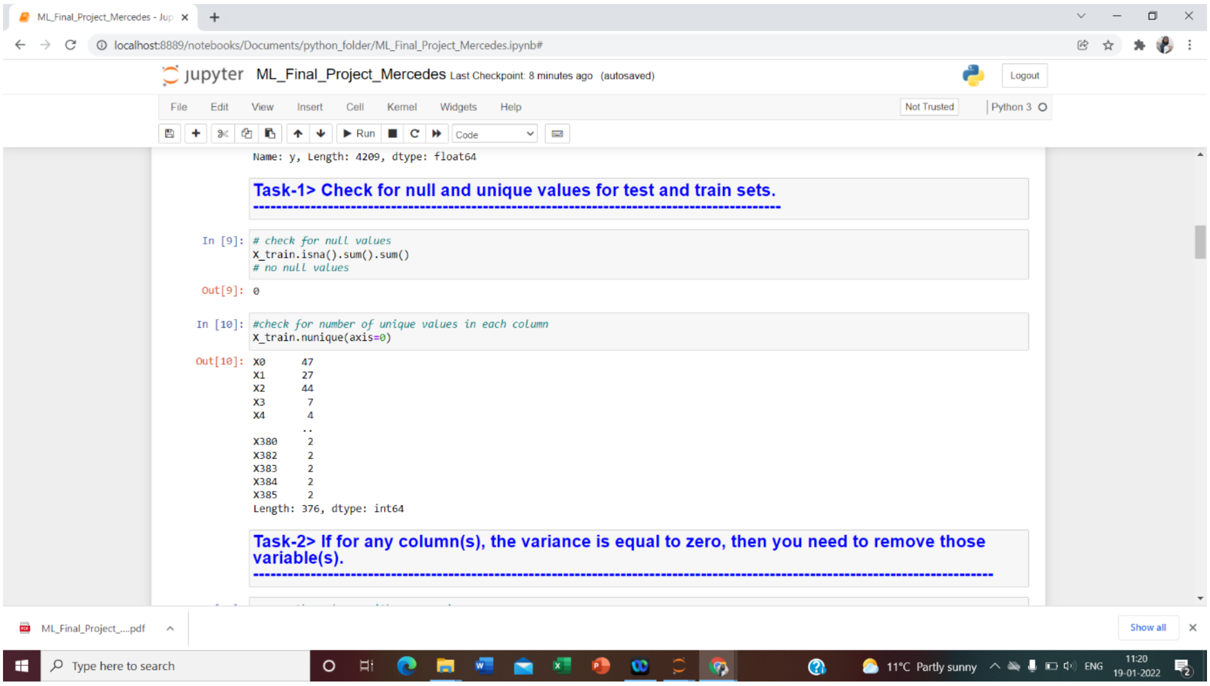
	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	k	v	at	a	d	u	j	o	0	0	...	0	0	1	0	0	0	0	0	0	0
1	k	t	av	e	d	y	l	o	0	0	...	1	0	0	0	0	0	0	0	0	0
2	az	w	n	c	d	x	j	x	0	0	...	0	0	0	0	0	0	1	0	0	0
3	az	t	n	f	d	x	l	e	0	0	...	0	0	0	0	0	0	0	0	0	0
4	az	v	n	f	d	h	d	n	0	0	...	0	0	0	0	0	0	0	0	0	0
5	t	b	e	c	d	g	h	s	0	0	...	0	0	1	0	0	0	0	0	0	0
6	al	r	e	f	d	f	h	s	0	0	...	0	0	0	0	0	0	0	0	0	0
7	o	l	as	f	d	f	j	a	0	0	...	0	0	0	0	0	0	0	0	0	0
8	w	s	as	e	d	f	i	h	0	0	...	1	0	0	0	0	0	0	0	0	0
9	j	b	aq	c	d	f	a	e	0	0	...	0	0	1	0	0	0	0	0	0	0

10 rows × 376 columns

In [8]:
```
y_train = train_df['y']
y_train
```

Out[8]:
```
0      130.81
1       88.53
2       76.26
3       80.62
4       78.02
```

3- Check for Null and Unique Values



jupyter ML_Final_Project_Mercedes Last Checkpoint: 8 minutes ago (autosaved)

```
Name: y, Length: 4209, dtype: float64
```

Task-1> Check for null and unique values for test and train sets.

In [9]:
```
# check for null values
X_train.isna().sum().sum()
# no null values
```

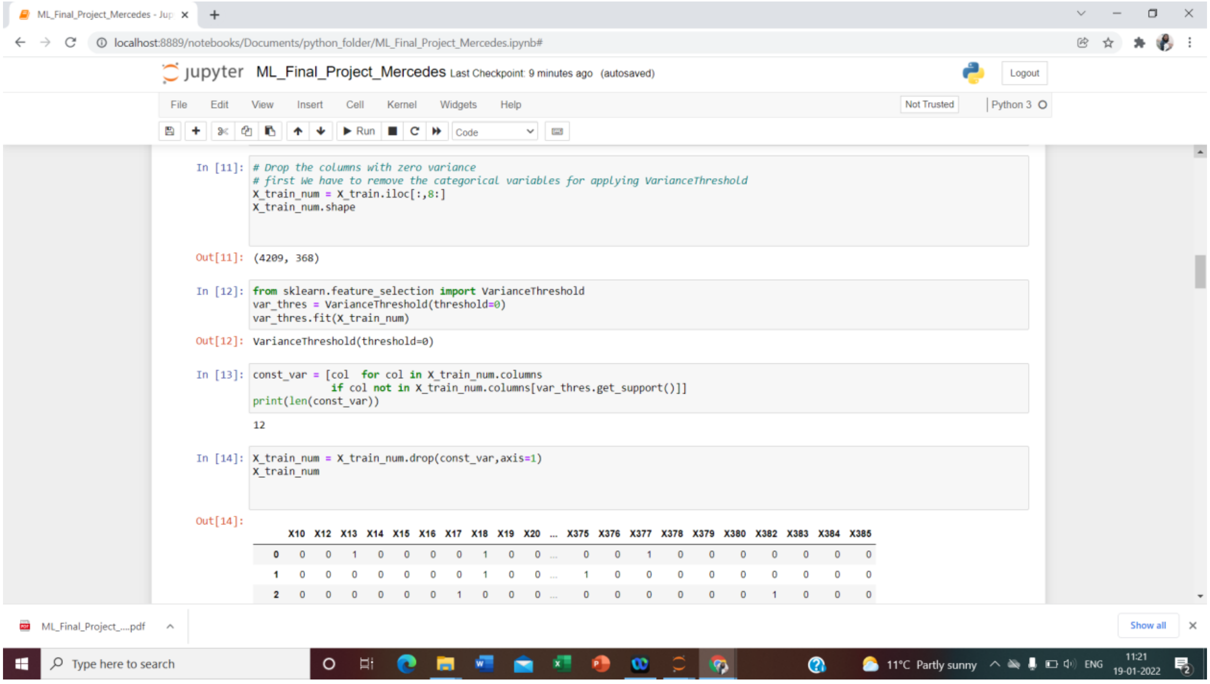
Out[9]:
```
0
```

In [10]:
```
#check for number of unique values in each column
X_train.nunique(axis=0)
```

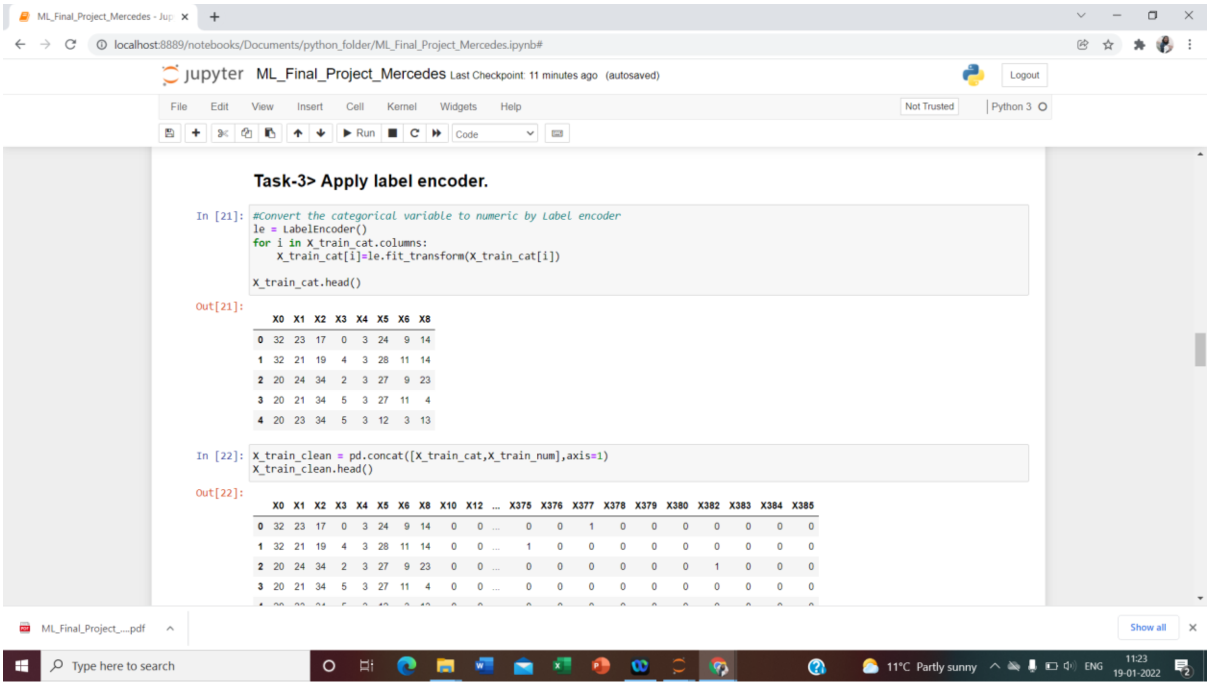
Out[10]:
```
X0      47
X1      27
X2      44
X3       7
X4       4
        ..
X380     2
X382     2
X383     2
X384     2
X385     2
Length: 376, dtype: int64
```

Task-2> If for any column(s), the variance is equal to zero, then you need to remove those variable(s).

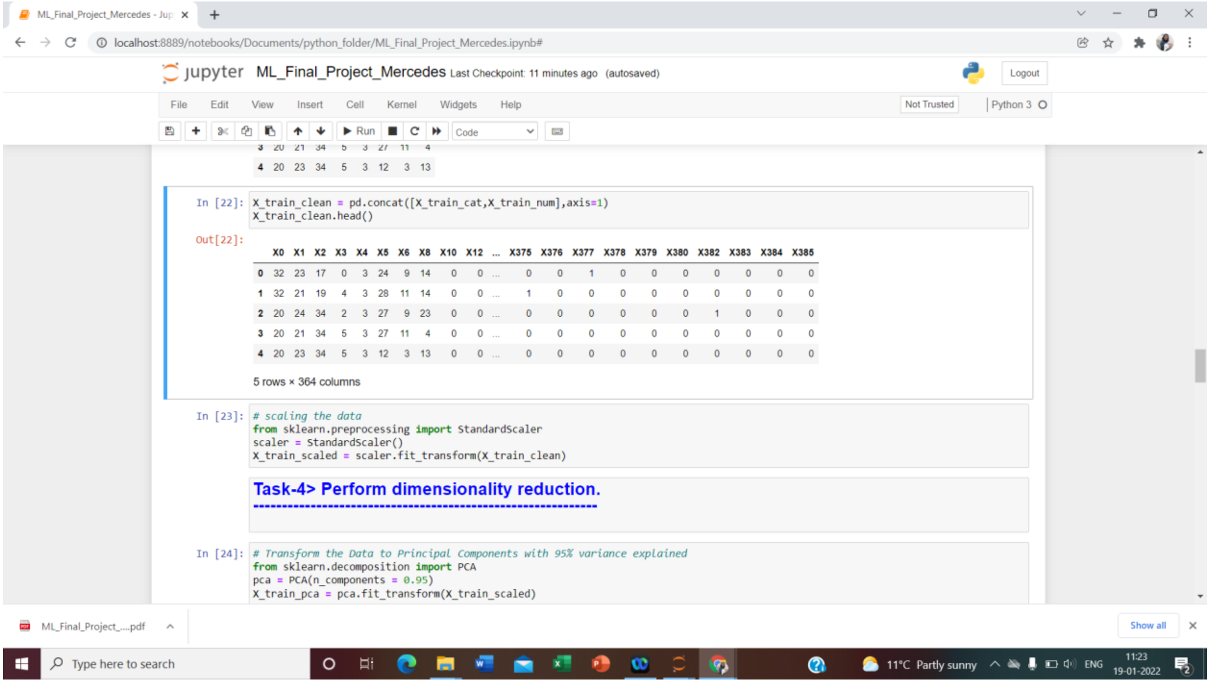
4- Drop the Zero variance Columns from Numeric Data



5- Apply LabelEncoder to the Categorical variable



6- Concatenate the Cleaned Numeric data and the Converted Categorical data



jupyter ML_Final_Project_Mercedes Last Checkpoint: 11 minutes ago (autosaved)

3	20	21	34	5	3	27	11	4
4	20	23	34	5	3	12	3	13

```
In [22]: X_train_clean = pd.concat([X_train_cat,X_train_num],axis=1)
         X_train_clean.head()
```

Out[22]:

	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

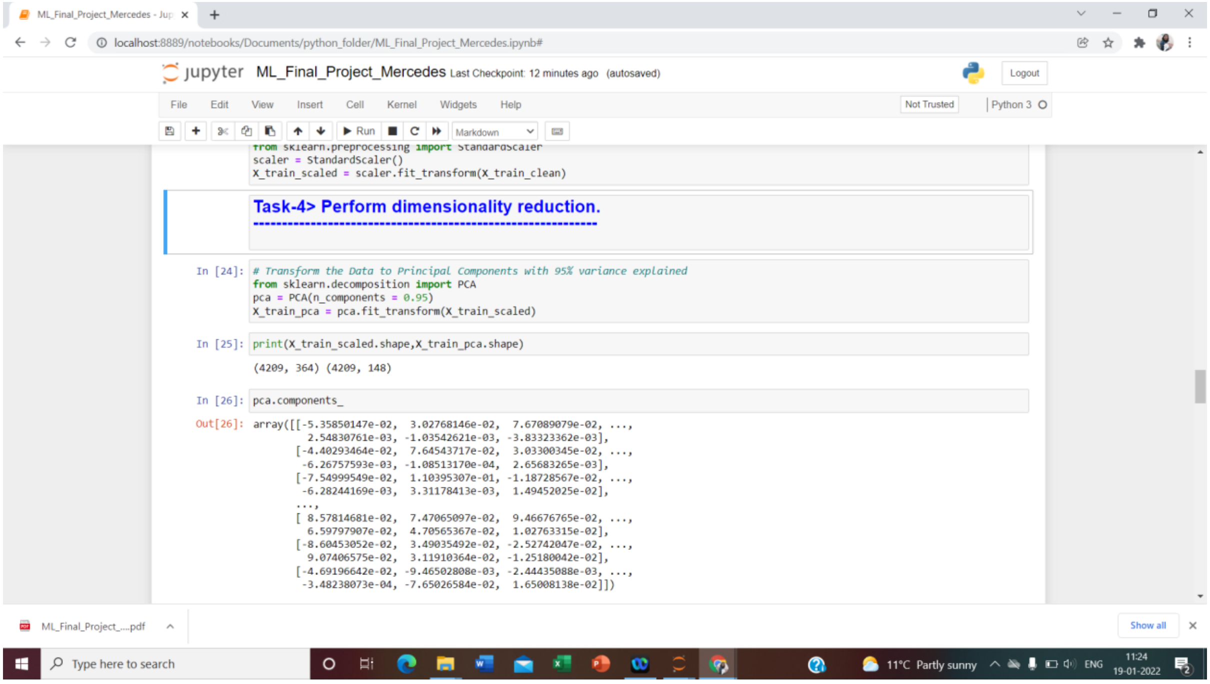
5 rows × 364 columns

```
In [23]: # scaling the data
         from sklearn.preprocessing import StandardScaler
         scaler = StandardScaler()
         X_train_scaled = scaler.fit_transform(X_train_clean)
```

Task-4> Perform dimensionality reduction.

```
In [24]: # Transform the Data to Principal Components with 95% variance explained
         from sklearn.decomposition import PCA
         pca = PCA(n_components = 0.95)
         X_train_pca = pca.fit_transform(X_train_scaled)
```

7- After Scaling Apply Dimensionality Reduction by using PCA



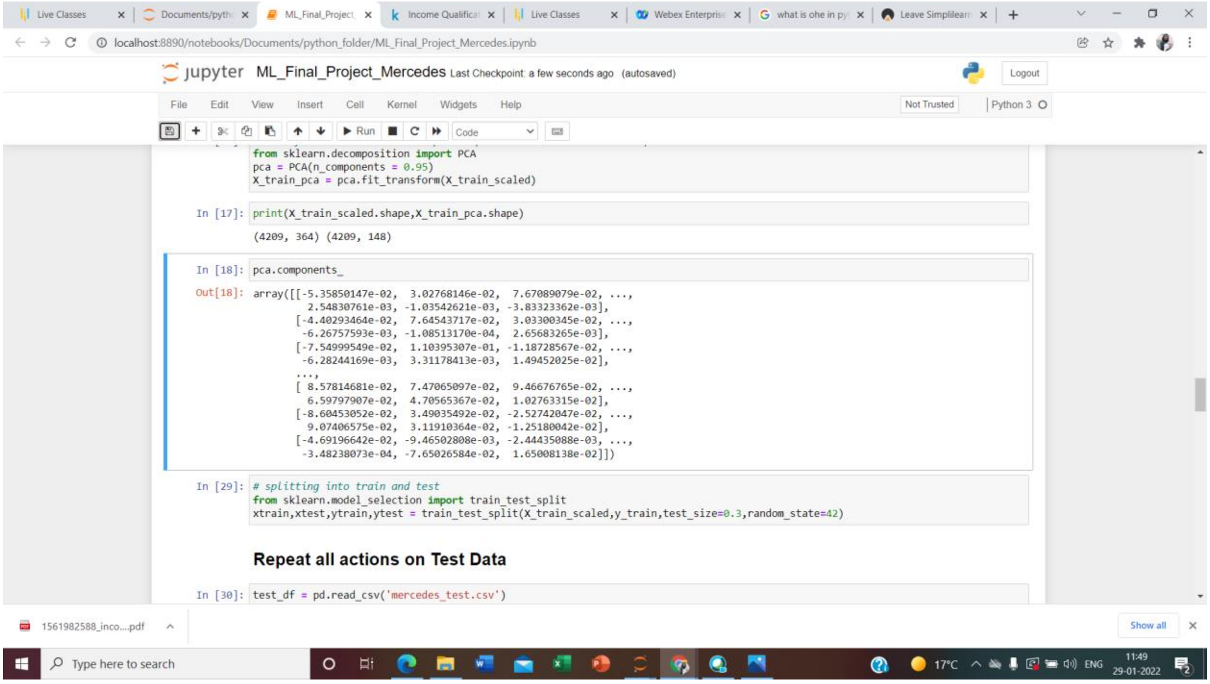
jupyter ML_Final_Project_Mercedes Last Checkpoint: 12 minutes ago (autosaved)

```
         from sklearn.preprocessing import StandardScaler
         scaler = StandardScaler()
         X_train_scaled = scaler.fit_transform(X_train_clean)
```

Task-4> Perform dimensionality reduction.

```
In [24]: # Transform the Data to Principal Components with 95% variance explained
         from sklearn.decomposition import PCA
         pca = PCA(n_components = 0.95)
         X_train_pca = pca.fit_transform(X_train_scaled)

In [25]: print(X_train_scaled.shape,X_train_pca.shape)

         (4209, 364) (4209, 148)

In [26]: pca.components_

Out[26]: array([[-5.35850147e-02,  3.02768146e-02,  7.67089079e-02, ...,
                  2.54830761e-03, -1.03542621e-03, -3.83323362e-03],
                [-4.40293464e-02,  7.64543717e-02,  3.03300345e-02, ...,
                 -6.26757593e-03, -1.08513170e-04,  2.65683265e-03],
                [-7.54999549e-02,  1.10395307e-01, -1.18728567e-02, ...,
                 -6.28244169e-03,  3.31178413e-03,  1.49452025e-02],
                ...,
                [ 8.57814681e-02,  7.47065097e-02,  9.46676765e-02, ...,
                  6.59797907e-02,  4.70565367e-02,  1.02763315e-02],
                [-8.60453052e-02,  3.49035492e-02, -2.52742047e-02, ...,
                  9.07406575e-02,  3.11910364e-02, -1.25180042e-02],
                [-4.69196642e-02, -9.46502808e-03, -2.44435088e-03, ...,
                 -3.48238073e-04, -7.65026584e-02,  1.65008138e-02]])
```

8- Apply Train Test Split



jupyter ML_Final_Project_Mercedes Last Checkpoint: a few seconds ago (autosaved)

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 0.95)
X_train_pca = pca.fit_transform(X_train_scaled)
```

In [17]:
```
print(X_train_scaled.shape,X_train_pca.shape)
```

```
(4209, 364) (4209, 148)
```

In [18]:
```
pca.components_
```

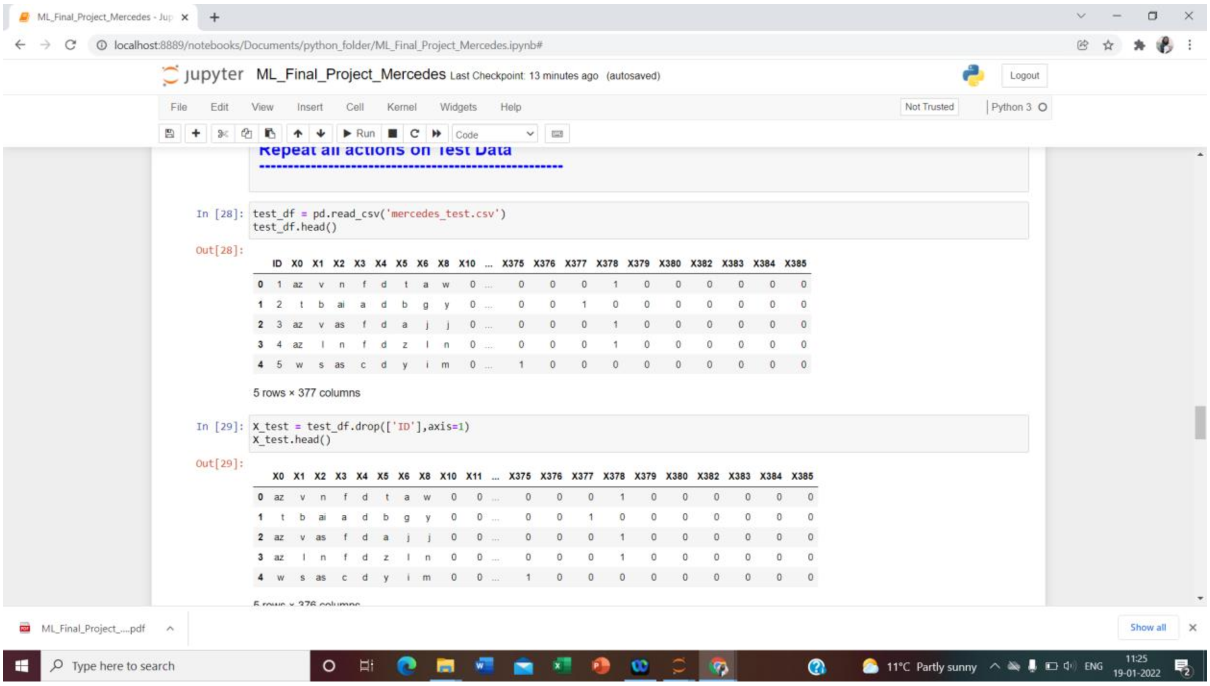
Out[18]:
```
array([[-5.35850147e-02,  3.02768146e-02,  7.67089079e-02, ...,
         2.54830761e-03, -1.03542621e-03, -3.83323362e-03],
       [-4.40293464e-02,  7.64543717e-02,  3.03300345e-02, ...,
        -6.26757593e-03, -1.08513170e-04,  2.65683265e-03],
       [-7.54999549e-02,  1.10395307e-01, -1.18728567e-02, ...,
        -6.28244169e-03,  3.31178413e-03,  1.49452025e-02],
       ...,
       [ 8.57814681e-02,  7.47065097e-02,  9.46676765e-02, ...,
         6.59797907e-02,  4.70565367e-02,  1.02763315e-02],
       [-8.60453052e-02,  3.49035492e-02, -2.52742047e-02, ...,
         9.07406575e-02,  3.11910364e-02, -1.25180042e-02],
       [-4.69196642e-02, -9.46502808e-03, -2.44435088e-03, ...,
        -3.48238073e-04, -7.65026584e-02,  1.65008138e-02]])
```

In [29]:
```
# splitting into train and test
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest = train_test_split(X_train_scaled,y_train,test_size=0.3,random_state=42)
```

Repeat all actions on Test Data

In [30]:
```
test_df = pd.read_csv('mercedes_test.csv')
```

9- Load Test Dataset



jupyter ML_Final_Project_Mercedes Last Checkpoint: 13 minutes ago (autosaved)

Repeat all actions on Test Data

In [28]:
```
test_df = pd.read_csv('mercedes_test.csv')
test_df.head()
```

Out[28]:

	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

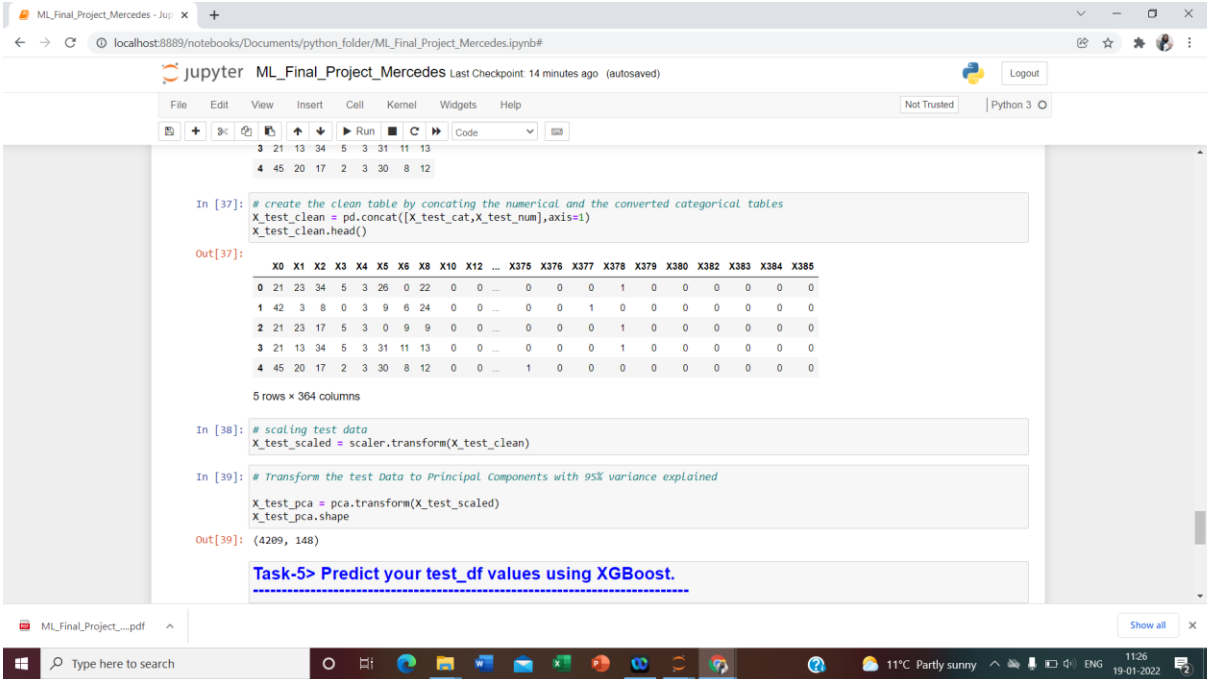
In [29]:
```
X_test = test_df.drop(['ID'],axis=1)
X_test.head()
```

Out[29]:

	X0	X1	X2	X3	X4	X5	X6	X8	X10	X11	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	az	v	n	f	d	t	a	w	0	0	...	0	0	0	1	0	0	0	0	0	0
1	t	b	ai	a	d	b	g	y	0	0	...	0	0	1	0	0	0	0	0	0	0
2	az	v	as	f	d	a	j	j	0	0	...	0	0	0	1	0	0	0	0	0	0
3	az	l	n	f	d	z	l	n	0	0	...	0	0	0	1	0	0	0	0	0	0
4	w	s	as	c	d	y	i	m	0	0	...	1	0	0	0	0	0	0	0	0	0

5 rows × 376 columns

10- Repeat all actions to get Clean Test Dataset



```
In [37]: # create the clean table by concating the numerical and the converted categorical tables
X_test_clean = pd.concat([X_test_cat,X_test_num],axis=1)
X_test_clean.head()
```

Out[37]:

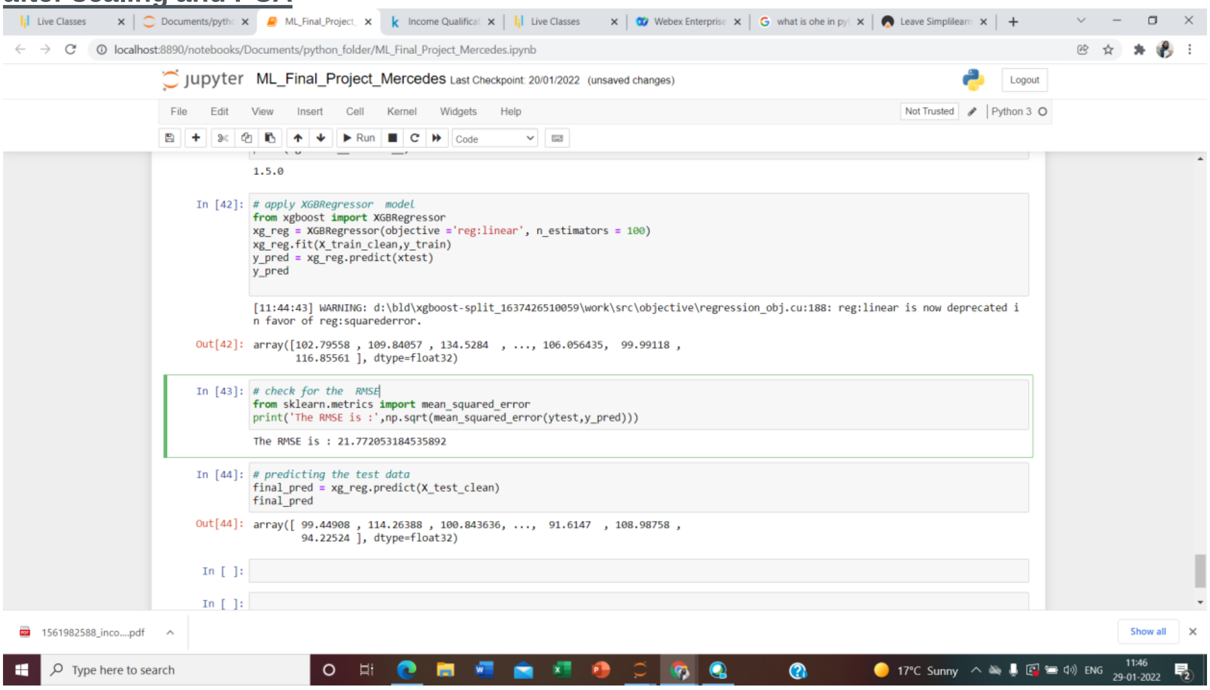
	X0	X1	X2	X3	X4	X5	X6	X8	X10	X12	...	X375	X376	X377	X378	X379	X380	X382	X383	X384	X385
0	21	23	34	5	3	26	0	22	0	0	...	0	0	0	1	0	0	0	0	0	0
1	42	3	8	0	3	9	6	24	0	0	...	0	0	1	0	0	0	0	0	0	0
2	21	23	17	5	3	0	9	9	0	0	...	0	0	0	1	0	0	0	0	0	0
3	21	13	34	5	3	31	11	13	0	0	...	0	0	0	1	0	0	0	0	0	0
4	45	20	17	2	3	30	8	12	0	0	...	1	0	0	0	0	0	0	0	0	0

5 rows × 364 columns

```
In [38]: # scaling test data
X_test_scaled = scaler.transform(X_test_clean)

In [39]: # Transform the test Data to Principal Components with 95% variance explained

X_test_pca = pca.transform(X_test_scaled)
X_test_pca.shape

Out[39]: (4209, 148)
```

Task-5> Predict your test_df values using XGBoost.

11- Check the RMSE and Predict the data by applying XGBoost to the Clean Test Dataset after scaling and PCA



```
1.5.0

In [42]: # apply XGBRegressor  model
from xgboost import XGBRegressor
xg_reg = XGBRegressor(objective ='reg:linear', n_estimators = 100)
xg_reg.fit(X_train_clean,y_train)
y_pred = xg_reg.predict(xtest)
y_pred

[11:44:43] WARNING: d:\bld\xgboost-split_1637426510059\work\src\objective\regression_obj.cu:188: reg:linear is now deprecated i
n favor of reg:squarederror.

Out[42]: array([102.79558 , 109.84057 , 134.5284  , ..., 106.056435,  99.99118 ,
        116.85561 ], dtype=float32)

In [43]: # check for the  RMSE
from sklearn.metrics import mean_squared_error
print('The RMSE is :',np.sqrt(mean_squared_error(ytest,y_pred)))

The RMSE is : 21.772053184535892

In [44]: # predicting the test data
final_pred = xg_reg.predict(X_test_clean)
final_pred

Out[44]: array([ 99.44908 , 114.26388 , 100.843636, ...,  91.6147  , 108.98758 ,
         94.22524 ], dtype=float32)
```