Name: Karthikeyan.S, Teena Mary
Reg: 1948013, 1948058
Class: 2MDS

A MACHINE LEARNING SOLUTION FOR REDUCING THE TIME CAR TAKES ON A TEST BENCH

1. PROBLEM

i. Description OR data story

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 car makers. 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 and every 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.

ii. Data Description:

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.

iii. Objective

1. Reduce the time that cars spend on the test bench.

2. To speedier testing, resulting in lower carbon dioxide emissions without reducing Daimler's standards.

2. MACHINE LEARNING PROBLEM

i. Type of Machine Learning Problem

We need to Reduce the time that cars spend on the test bench.

ii. Train and Test Construction

We build train and test by randomly splitting in the ratio of 70:30 or 80:20 whatever we choose as we have sufficient points to work with.

Name: Karthikeyan.S, Teena Mary
Reg: 1948013, 1948058
Class: 2MDS

3. EXPLORATORY DATA ANALYSIS

i. Importing important libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
.
.
.
from sklearn.linear_model import
LinearRegression
import scipy.stats as stats
from sklearn.externals import joblib
```

ii. Reading data and basic stats

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

Target Variable:

- "y" is the variable we need to predict. So, let us do some analysis on this variable first.
- Variable y is of type float
- X0, X1, X2, X3, X4, X5, X6, X8 are of type object
- Rest of the columns are int type
- We will convert [X0, X1, X2, X3, X4, X5, X6, X8] to categorical types and plot to see the distribution of values.

Name: Karthikeyan.S, Teena Mary
Reg: 1948013, 1948058
Class: 2MDS

iii. Checking for missing values

```
def check_missing_values(df):

    if df.isnull().any().any():
        print("There are missing values in the data")
    else:
        print("There are no missing values in the data")

check_missing_values(train_df)
check_missing_values(test_df)
```

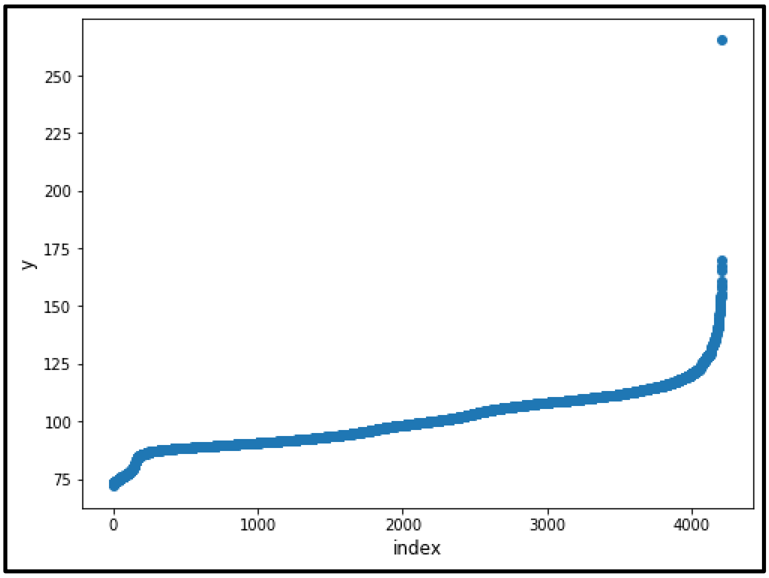
Once we run our data in the above function and check if there are any missing values, we obtain a positive result that there aren't any missing values.

iv. Plotting

Here we plot the values of Y to check if there are any outliers and if any, we will remove them by normalization technique.

```
plt.figure(figsize=(8,6))
plt.scatter(range(train_df.shape[0]),
np.sort(train_df.y.values))
plt.xlabel('index', fontsize=12)
plt.ylabel('y', fontsize=12)
plt.show()
```

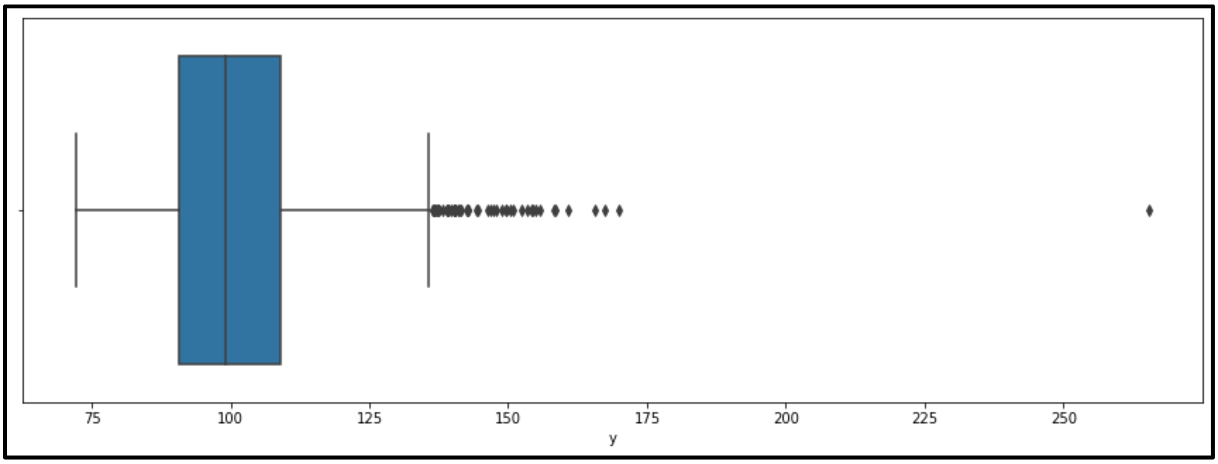
v. Plotting y values



Name: Karthikeyan.S, Teena Mary
Reg: 1948013, 1948058
Class: 2MDS

Here, we have observed one outlier at approximately 260. We can also check with the box plot for cross reference.

```
plt.figure(figsize=(15,5))
sns.boxplot(train_df.loc[:,'y'])
plt.show()
```



Now we use the method of Z score normalization to remove the extreme value and we use a threshold value of 10 as it fits the data well.

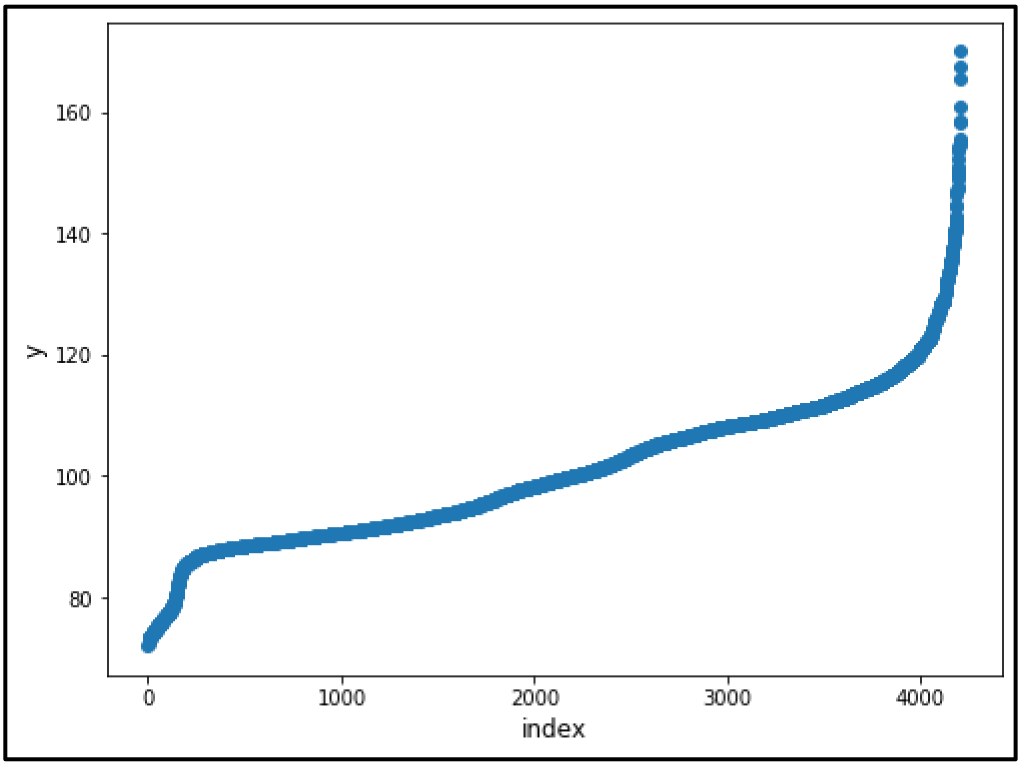
Once we normalize the data we will check with the plots if any more outliers are still present.

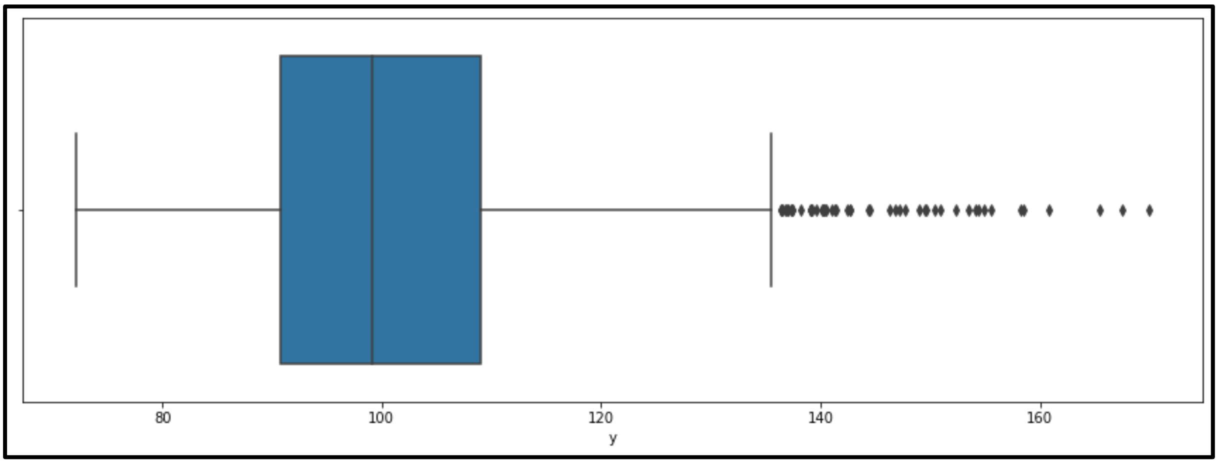
```
train_df['x'] = np.abs(stats.zscore(train_df.loc[:,'y']))
outlier_ids = train_df[train_df['x']>10].ID
train_df_final = train_df[~train_df['ID'].isin(list(outlier_ids))]

plt.figure(figsize=(8,6))
plt.scatter(range(train_df_final.shape[0]),
np.sort(train_df_final.y.values))
plt.xlabel('index', fontsize=12)
plt.ylabel('y', fontsize=12)
plt.show()

plt.figure(figsize=(15,5))
sns.boxplot(train_df_final.loc[:,'y'])
plt.show()
```

Name: Karthikeyan.S, Teena Mary
Reg: 1948013, 1948058
Class: 2MDS





Hence through the plots we can confirm that the outliers have disappeared.

vi. Data type of all the variables present in the dataset.

```
dtype_df = train_df_final.dtypes.reset_index()
dtype_df.columns = ["Count", "Column Type"]
dtype_df.groupby("Column
Type").aggregate('count').reset_index()
```

	Column Type	Count
0	int64	369
1	float64	1
2	object	8

Maximum of the columns are integers.

8 categorical columns.

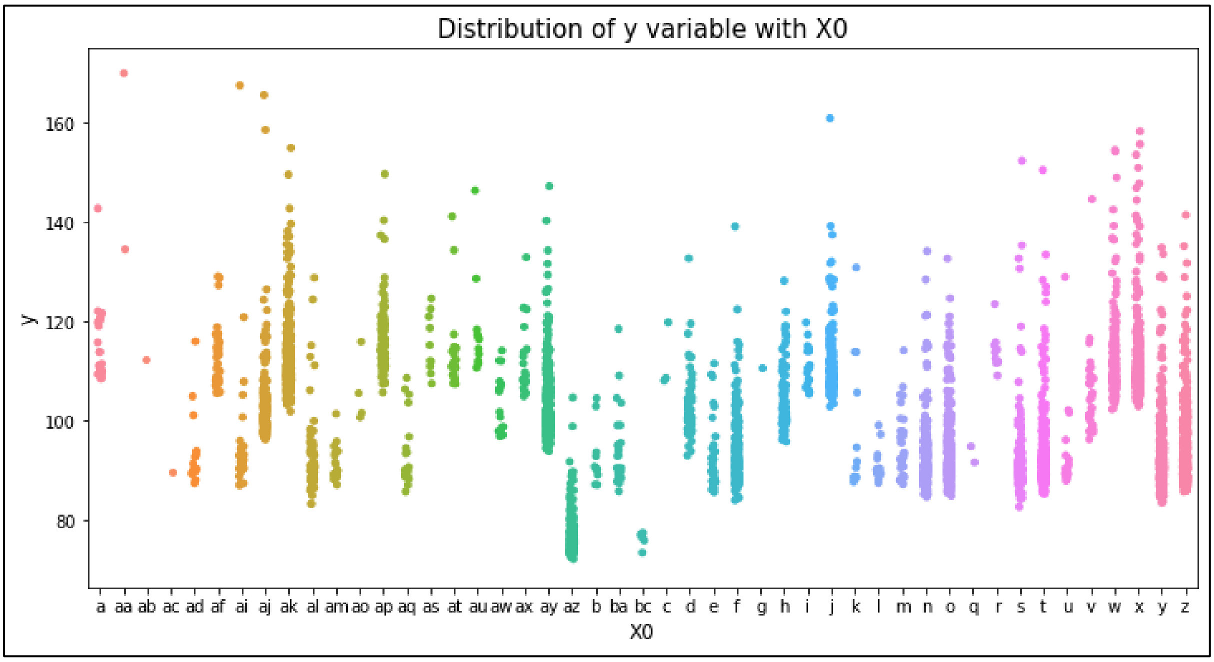
1 float column (target variable) i.e. 'y'

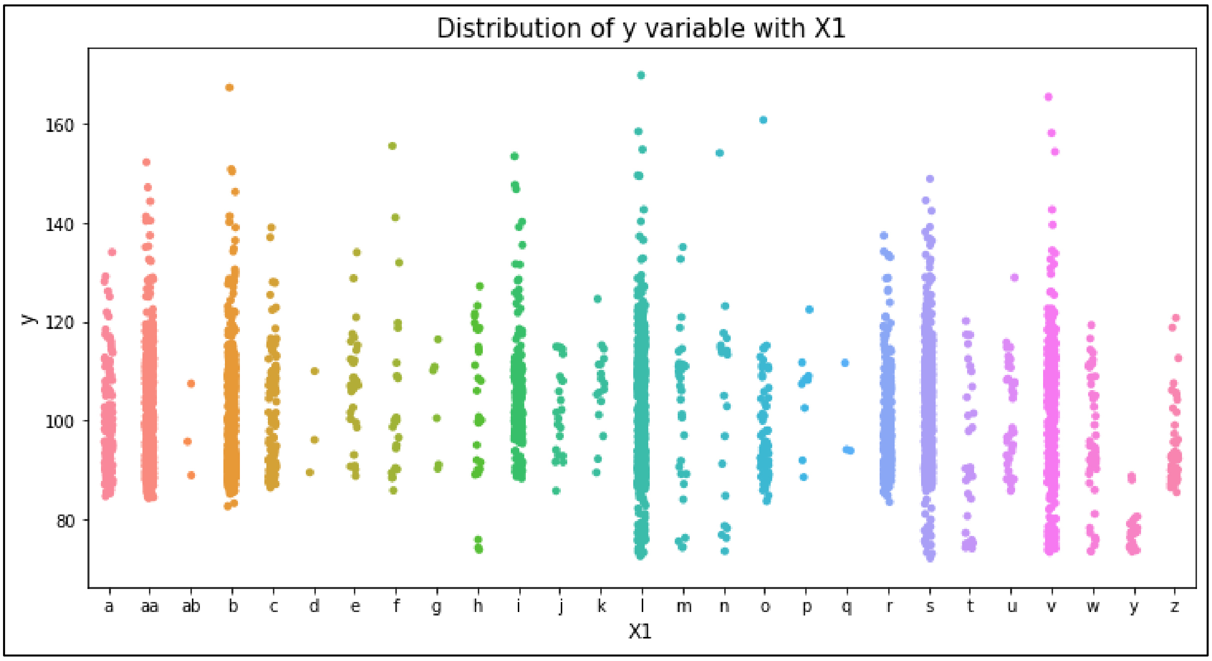
X0 to X8 are the categorical columns by observing the dataset.

vii. Plotting these categorical Values

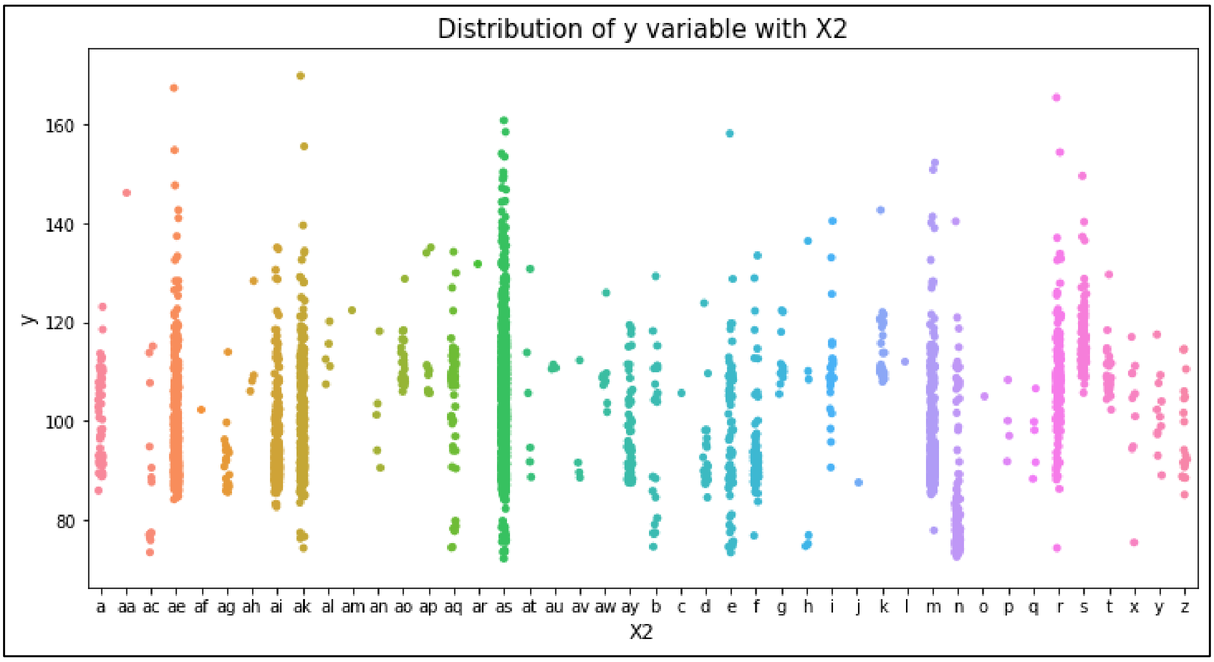
```
var_name = ['X0','X1','X2','X3','X4','X5','X6','X8']
for val in var_name:
    col_order =
np.sort(train_df_final[val].unique()).tolist()
    plt.figure(figsize=(12,6))
    sns.stripplot(x=val, y='y', data=train_df_final,
order=col_order)
    plt.xlabel(val, fontsize=12)
    plt.ylabel('y', fontsize=12)
    plt.title("Distribution of y variable with "+val,
fontsize=15)
    plt.show()
```

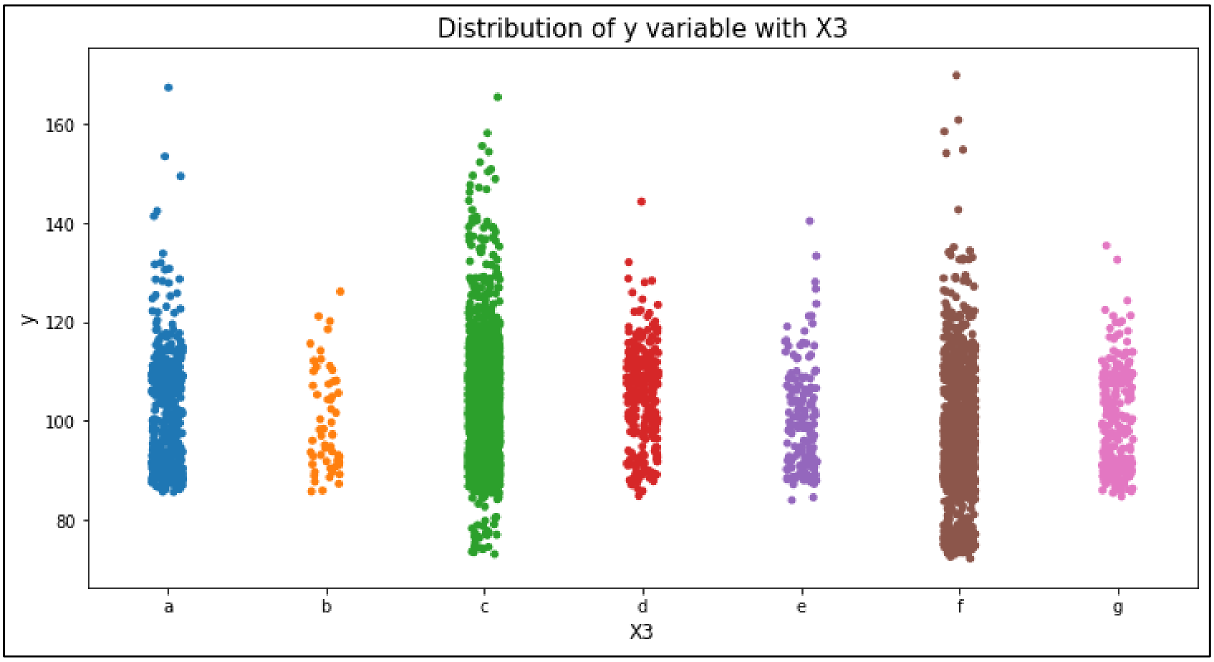
Name: Karthikeyan.S, Teena Mary
Reg: 1948013, 1948058
Class: 2MDS



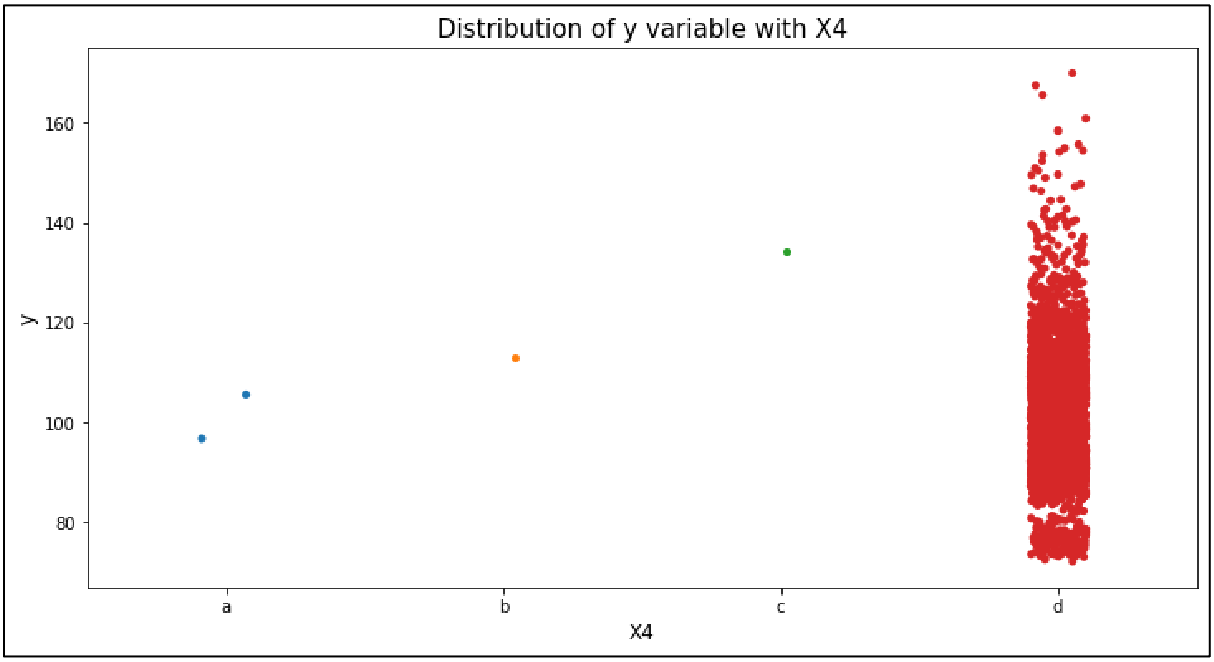


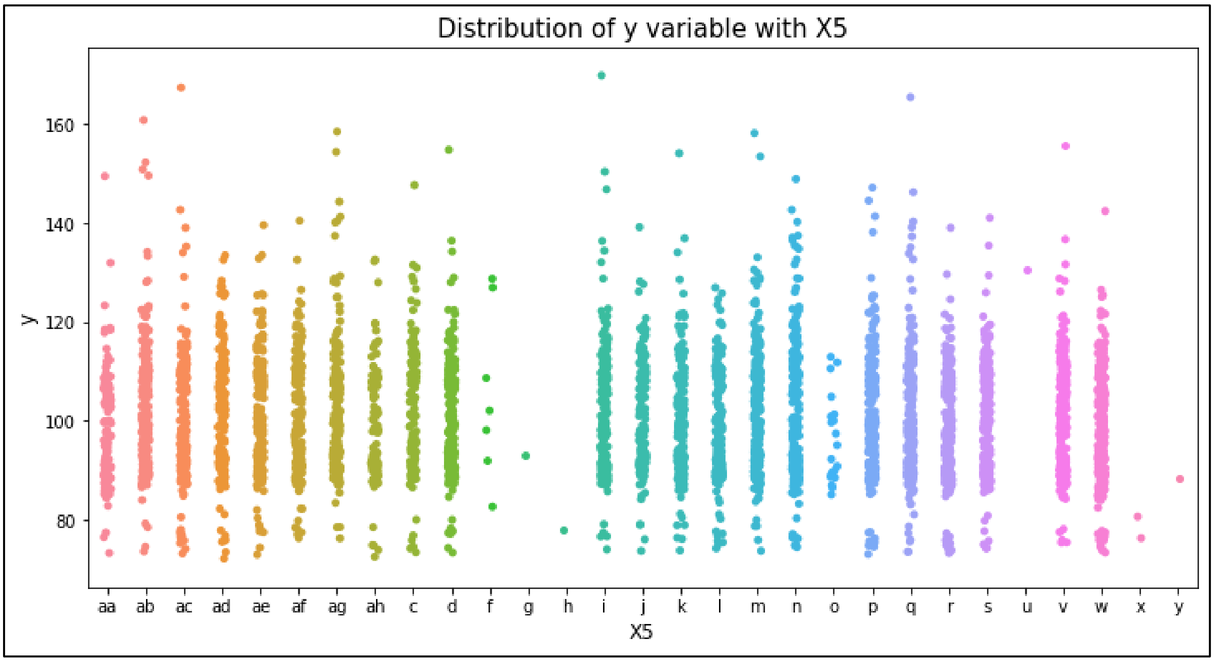
Name: Karthikeyan.S, Teena Mary
Reg: 1948013, 1948058
Class: 2MDS



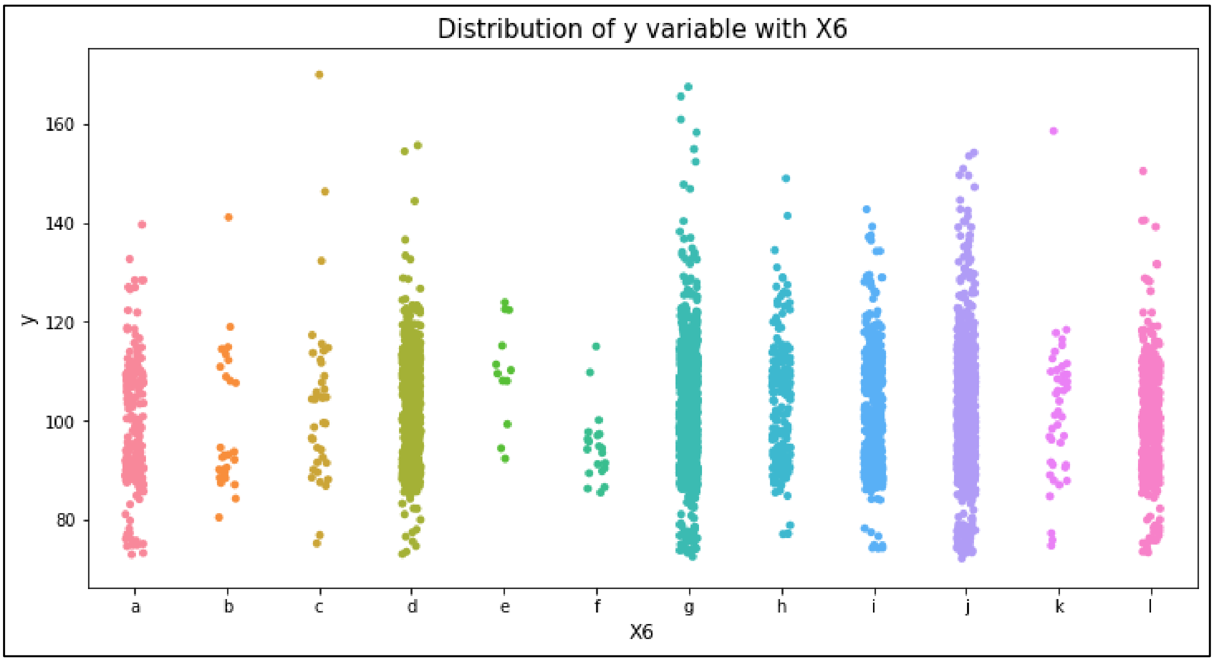


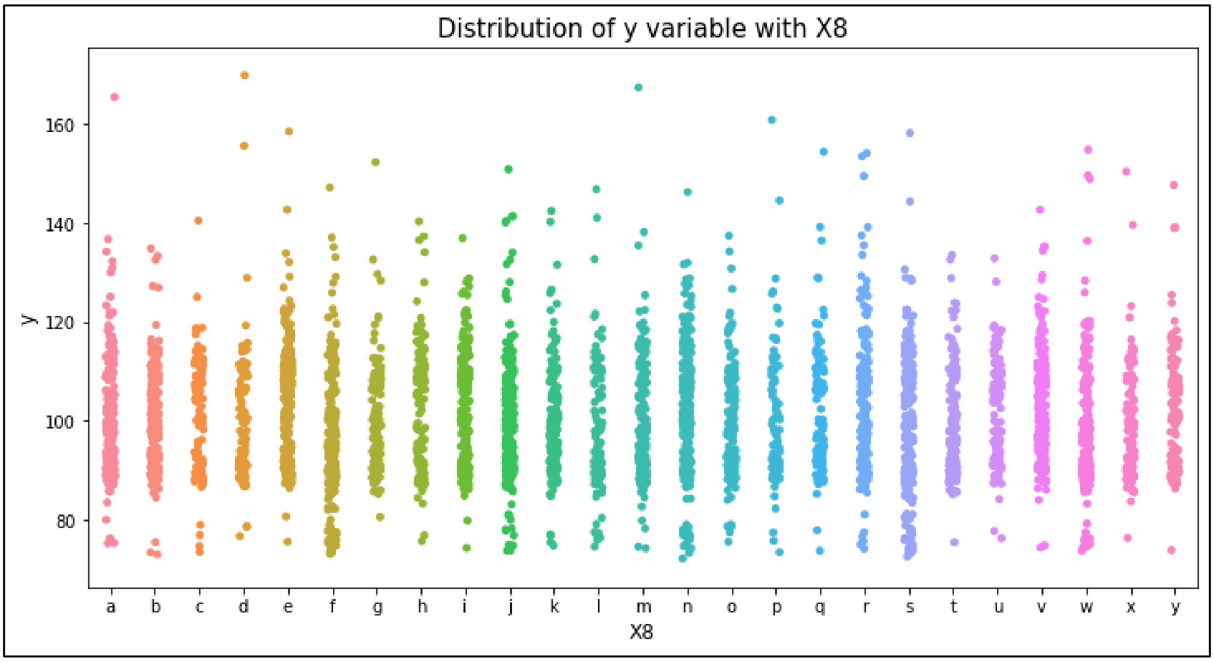
Name: Karthikeyan.S, Teena Mary
Reg: 1948013, 1948058
Class: 2MDS





Name: Karthikeyan.S, Teena Mary
Reg: 1948013, 1948058
Class: 2MDS





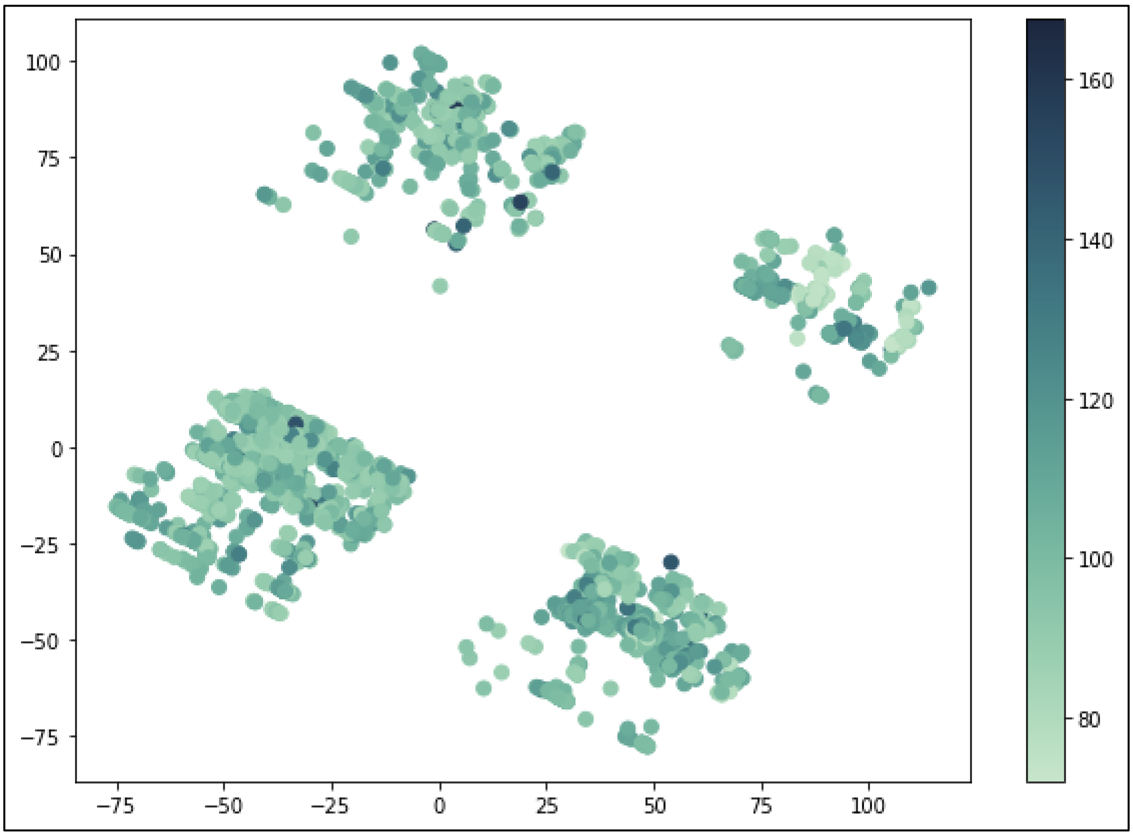
- We have observed that X0, X1, X2, X5, X6 and X8 have larger data point.
- X4 and X3 have lesser data point.

Name: Karthikeyan.S, Teena Mary
Reg: 1948013, 1948058
Class: 2MDS

4. MACHINE LEARNING TECHNIQUES

i. Principal component analysis - PCA

```
pca = PCA(n_components=2)
pca_data = pca.fit_transform(X_train)
cmap = sns.cubehelix_palette(as_cmap=True,rot=-.4)
f, ax = plt.subplots(figsize=(10,7))
points = ax.scatter(pca_data[:,0], pca_data[:,1], c=y_train, s=50, cmap=cmap)
f.colorbar(points)
plt.show()
```



Here we can see how PCA visualize all the data point far separated from each other, they are not forming tightly group.

Name: Karthikeyan.S, Teena Mary
Reg: 1948013, 1948058
Class: 2MDS

ii. K-Nearest Neighbours Regressor

```
knn = KNeighborsRegressor(n_neighbors=2)

knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

r2_score_knn = round(r2_score(y_test, y_pred),3)#taking r2score
accuracy = round(knn.score(X_train, y_train) *100,2)#taking accuracy

results = {'r2_score':r2_score_knn, 'accuracy':accuracy}
print (results)
```

```
{'r2_score': 0.165, 'accuracy': 75.42}
```

iii. Support Vector Regressor

```
from sklearn.metrics import r2_score
clf = SVR()

clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

r2_score = round(r2_score(y_test, y_pred),3)#taking r2score
accuracy = round(clf.score(X_train, y_train) * 100, 2)

results = {'r2_score':r2_score, 'accuracy':accuracy}
print(results)
```

```
{'r2_score': -0.031, 'accuracy': -2.43}
```

Name: Karthikeyan.S, Teena Mary
Reg: 1948013, 1948058
Class: 2MDS

iv. Random Forest Regressor

```
from sklearn.metrics import r2_score
clf = RandomForestRegressor(n_estimators = 60 ,max_depth=5,oob_score=True)

clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

r2_score = round(r2_score(y_test, y_pred),3)#taking r2score
accuracy = round(clf.score(X_train, y_train) * 100, 2)

results = {'r2_score':r2_score, 'accuracy':accuracy}
print (results)
```

```
{'r2_score': 0.474, 'accuracy': 64.77}
```

v. Linear Regression

```
from sklearn.metrics import r2_score
clf = LinearRegression()

clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

r2_score = round(r2_score(y_test, y_pred),3)#taking r2score
accuracy = round(clf.score(X_train, y_train) * 100, 2)

results = {'r2_score':r2_score, 'accuracy':accuracy}
print (results)
```

```
{'r2_score': -240594149922716.0, 'accuracy': 63.06}
```

Name: Karthikeyan.S, Teena Mary
Reg: 1948013, 1948058
Class: 2MDS

vi. Checking the feature importance using the XGBoost.

XGBoost is one of the most popular machine learning algorithms these days. Regardless of the type of prediction task at hand; regression or classification.

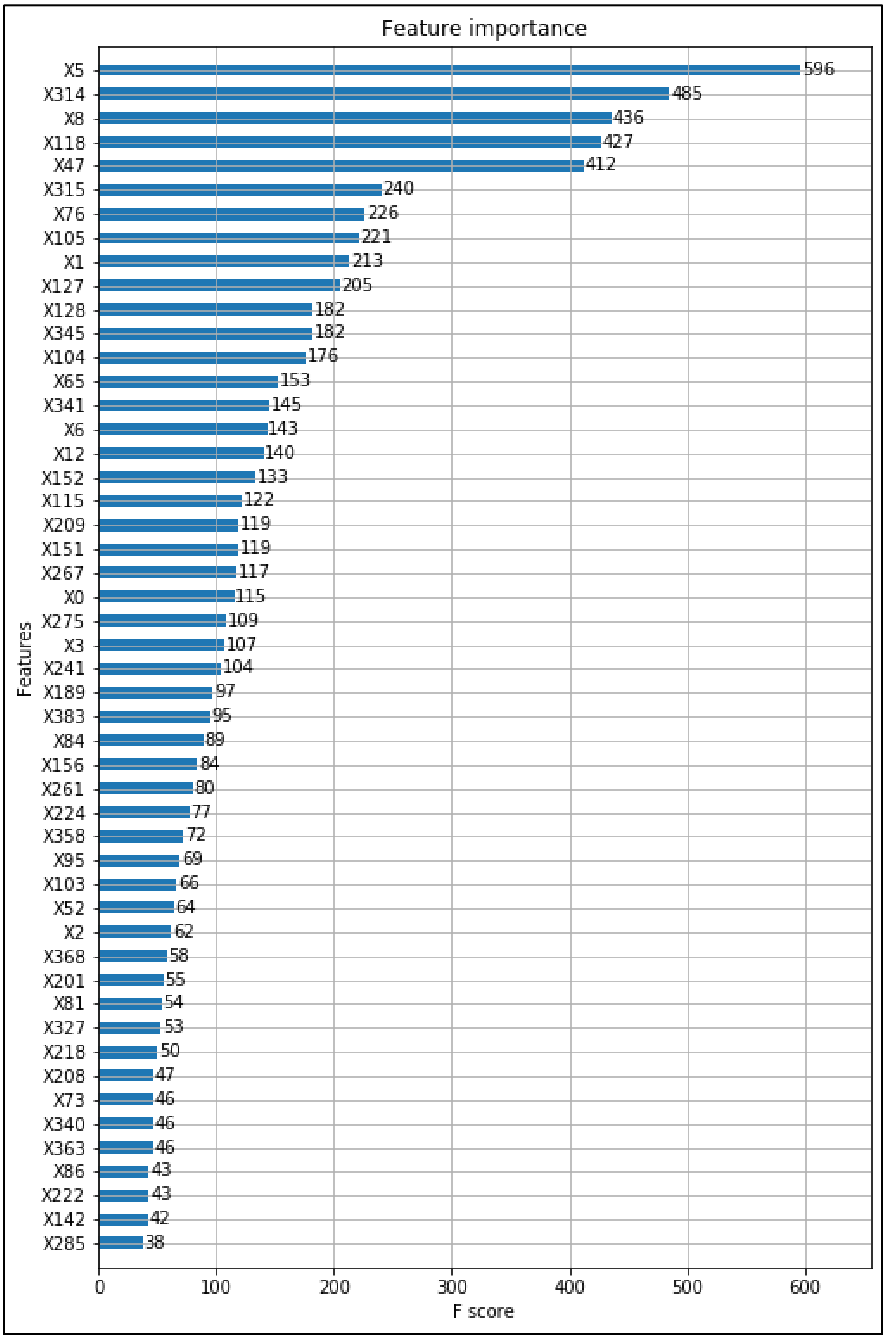
Here we are including some important commands involved in xgboost.

```
params = {

  'n_trees': 500,
   'eta': 0.005,
   'max_depth': 4,
   'subsample': 0.95,
   'objective': 'reg:linear',
   'eval_metric': 'rmse',
   'base_score': y_mean, # base prediction = mean(target)
   'silent': 1
}
cv_result = xgb.cv(params,
              d_train,
              num_boost_round,
              nfold = 3,
              early_stopping_rounds=50,
              feval=r2_score_metric,#here we have used our metric method
              verbose_eval=100,
              show_stdv=False
             )
```

This method helps us in constructing decision trees and boosting its performance so that it yields the best features with its importance level.

Name: Karthikeyan.S, Teena Mary
Reg: 1948013, 1948058
Class: 2MDS



So here we see that the features are extracted and plotted according to their importance.

The top 5 features are:

X5 -> 596

X8 -> 436

X1 -> 213

X6 -> 143

X0 -> 115

5. CONCLUSION

S.no	Model Algo	R^2 Score	Accuracy
1.	K-Nearest Neighbours Regressor	0.267	75.42
2.	Support Vector Regressor	0.208	-2.34
3.	Random Forest Regressor	0.476	64.77
3.	Linear Regression	-4.882113861562633e+16	63.06

- Linear Regression model not suit for this type of problem.

So here we can see the value of accuracy for K-Nearest Neighbours Regressor is higher and hence this machine learning technique is the best for our case.

References:

https://www.geeksforgeeks.org/scipy-stats-zscore-function-python/

https://www.kaggle.com/anokas/mercedes-eda-xgboost-starter-0-55

https://towardsdatascience.com/pca-using-python-scikit-learn-e653f8989e60

https://scikit-learn.org/stable/modules/svm.html

https://www.geeksforgeeks.org/linear-regression-python-implementation/