# SAFE DRIVING CHALLENGE

## ML Project Report

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE & ENGINEERING**

**SUBMITTED BY**


| NAME OF THE STUDENT | ROLL NO |
|---|---|
| Ms. M SUSMITHA | 17WH1A0591 |
| Ms. M NAGA PRAVALLIKA | 18WH5A0517 |
| Ms. T BINDHU BHARGAVI | 18WH5A0520 |


**Department of Computer Science and Engineering**

**BVRIT HYDERABAD**

**College of Engineering for Women**

**(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)**

**Bachupally, Hyderabad – 500090**

**Department of Computer Science and Engineering**
**BVRIT HYDERABAD**
**College of Engineering for Women**
**(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)**
**Bachupally, Hyderabad – 500090**

# Acknowledgement

Firstly, I would like to express my immense gratitude towards **BVRIT HYDERABAD College of Engineering for Women**, which created a great platform to attain profound technical skills in the field of Computer Science though this industry enabled learning WISE.

I would like to extend my sincere thanks and gratitude to **Dr. K V N Sunitha,** Principal, BVRIT HYDERABAD College of Engineering for Women and WISE team of college for their meticulous planning and conduction of this learning program.

I would also like to extend my sincere thanks to WISE & Team of Talentsprint for enabling us with this unique learning platform.

M Susmitha(17WH1A0591)

M Naga Pravallika(18WH5A0517)

T Bindhu Bhargavi(18WH5A0520)

# INDEX



# LIST OF FIGURES

# ABSTRACT

In this project we introduce a classifier which takes in multidimensional data consisting of real-world measurements of physical, environmental and vehicular continuous features obtained from number of driving sessions. We will show that using Naive Bayes classifier which assumes the data distribution to be Gaussian distribution we can make a prediction weather the driver is alerted or not while driving and achieve reasonable low misclassification rate for the given data. We will inspect how insight into relevant features were obtain by using Principal Component Analysis (PCA) and simple correlation matrix. We were able to obtain a misclassification rate as low as 12.03 % and 27.07 % for the test and training data respectively.

# INTRODUCTION

With a training and test set consisting of 33 features from real time measurements test we want to use that information to predict if a certain driver is alerted or not alerted while driving. Here our goal is to construct a binary classifier which will predict a binary target value using the whole or a subset of the 33 features and give a prediction as Predictions = ( 1 if the driver is alert 0 if the driver is not alert (1) A. Datasets The datasets are gained from the website www.kaggle.com and consist of one training set and one test set. The datasets include measurements from total of 510 real time driving session where each driving session takes 2 minutes. This gives a new measurement of the each of the 33 features every 100ms. The headers in the datasets are listed in table I below. The size of the training set is a measurement set of 510 driving sessions done by 100 people. This results in a 604330×33 as the size of the training set. The test data have fewer observations and has

# Problem Statement

Driving while distracted, fatigued or drowsy may lead to accidents. Activities that divert the driver's attention from the road ahead, such as engaging in a conversation with other passengers in the car, making or receiving phone calls, sending or receiving text messages, eating while driving or events outside the car may cause driver distraction. Fatigue and drowsiness can result from driving long hours or from lack of sleep.

The data for this Kaggle challenge shows the results of a number of "trials", each one representing about 2 minutes of sequential data that are recorded every 100ms during a driving session on the road or in a driving simulator. The trials are samples from some 100 drivers of both genders, and of different ages and ethnic backgrounds. The files are structured as follows:

The first column is the Trial ID - each period of around 2 minutes of sequential data has a unique trial ID. For instance, the first 1210 observations represent sequential observations every 100ms, and therefore all have the same trial ID The second column is the observation number - this is a sequentially increasing number within one trial ID The third column has a value X for each row where

X = 1 if the driver is alert

X = 0 if the driver is not alert

The next 8 columns with headers P1, P2 , …….., P8 represent physiological data;

The next 11 columns with headers E1, E2, …….., E11 represent environmental data;

The next 11 columns with headers V1, V2, …….., V11 represent vehicular data;

## APPROACH

- Initially, we have analyzed train and test datasets
- Imported the required libraries
- By using Data preprocessing, logistic regression, feature engineering, PCA, Support vector regression, Neural network we have predicted the output.

## STATISTICS OF THE CODE

- We have used google colaboratory to predict the output.

# CODE

SDC(ML).ipynb

File Edit View Insert Runtime Tools Help Cannot save changes

+ Code + Text Copy to Drive

## Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn import preprocessing
from sklearn.linear_model import LogisticRegressionCV
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_auc_score

from sklearn.svm import LinearSVC # Linear Support Vector Classigication
from sklearn.svm import NuSVC

from sklearn.model_selection import GridSearchCV, StratifiedKFold, train_test_split
```

```
[ ] train = pd.read_csv('/content/drive/My Drive/train.csv')
    test = pd.read_csv('/content/drive/My Drive/test.csv')
    exp = pd.read_csv('/content/drive/My Drive/example_submission.csv')
```

# FIRST MODEL

First Model

Data preprocessing

```
[ ] new_df = pd.DataFrame()
```

```
[ ] noalert = train.index[train['IsAlert'] == 0]
    turnnoalert = train.iloc[noalert-1][train['IsAlert'] == 1]
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:2: UserWarning: Boolean Series key will be reindexed to match DataFrame index.
```

```
[ ] for i in turnnoalert.index: # Return 500ms before and after the moment
        new_df = new_df.append(train[i-5:i+6])
```

```
[ ] new_df = new_df.drop_duplicates()
```

```
[ ] new_df
```

```
[ ] new_df
```

| | TrialID | ObsNum | IsAlert | P1 | P2 | P3 | P4 | P5 | P6 | P7 | P8 | E1 | E2 | E3 | E4 | E5 | E6 | E7 | E8 | E9 | E10 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **604323** | 510 | 1193 | 1 | 32.0303 | 7.68265 | 800 | 75.0000 | 0.081731 | 680 | 88.2353 | 0 | 17.807 | 222.11 | 0 | 0 | 0.016379 | 323 | 2 | 2 | 1 | 64 |
| **604324** | 510 | 1194 | 1 | 32.0051 | 10.13240 | 800 | 75.0000 | 0.081731 | 680 | 88.2353 | 0 | 17.807 | 222.11 | 0 | 0 | 0.016379 | 322 | 2 | 2 | 1 | 64 |
| **604325** | 510 | 1195 | 1 | 32.0393 | 12.45040 | 800 | 75.0000 | 0.081731 | 680 | 88.2353 | 0 | 17.807 | 222.11 | 0 | 0 | 0.016379 | 322 | 2 | 2 | 1 | 64 |
| **604326** | 510 | 1196 | 1 | 32.0762 | 10.06180 | 800 | 75.0000 | 0.081731 | 680 | 88.2353 | 0 | 17.807 | 222.11 | 0 | 0 | 0.016379 | 322 | 2 | 2 | 1 | 64 |
| **604327** | 510 | 1197 | 1 | 32.1154 | 17.84500 | 800 | 75.0000 | 0.081731 | 680 | 88.2353 | 0 | 17.807 | 222.11 | 0 | 0 | 0.016379 | 322 | 2 | 2 | 1 | 64 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **604314** | 510 | 1184 | 0 | 31.9667 | 9.17322 | 688 | 87.2093 | 0.089515 | 684 | 87.7193 | 0 | 17.807 | 222.11 | 0 | 0 | 0.016379 | 327 | 2 | 2 | 1 | 64 |
| **604315** | 510 | 1185 | 0 | 31.9291 | 15.50140 | 688 | 87.2093 | 0.089515 | 684 | 87.7193 | 0 | 17.807 | 222.11 | 0 | 0 | 0.016379 | 327 | 2 | 2 | 1 | 64 |
| **604316** | 510 | 1186 | 0 | 31.9820 | 8.30669 | 688 | 87.2093 | 0.089515 | 684 | 87.7193 | 0 | 17.807 | 222.11 | 0 | 0 | 0.016379 | 327 | 2 | 2 | 1 | 64 |
| **604317** | 510 | 1187 | 0 | 31.9516 | 9.91952 | 688 | 87.2093 | 0.089515 | 684 | 87.7193 | 0 | 17.807 | 222.11 | 0 | 0 | 0.016379 | 327 | 2 | 2 | 1 | 64 |
| **604318** | 510 | 1188 | 1 | 31.9446 | 12.47740 | 688 | 87.2093 | 0.089515 | 684 | 87.7193 | 0 | 17.807 | 222.11 | 0 | 0 | 0.016379 | 323 | 2 | 2 | 1 | 64 |

37421 rows × 33 columns

## Feature Engineering

```
[ ] pcadata = new_df.drop(columns = ['TrialID', 'ObsNum', 'IsAlert']) # Drop unneccessary columns
```

## Standarization

```
[ ] X_scaled = preprocessing.scale(pcadata)
    pcadata.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 37421 entries, 604323 to 604318
Data columns (total 30 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   P1      37421 non-null  float64
 1   P2      37421 non-null  float64
 2   P3      37421 non-null  int64
 3   P4      37421 non-null  float64
 4   P5      37421 non-null  float64
```

## Principle Component Analysis

```
[ ] pca=PCA()
    pca.fit(X_scaled)
    X_pca=pca.transform(X_scaled)
```

```
[ ] #let's check the shape of X_pca array
    print ("shape of X_pca", X_pca.shape)
```

```
shape of X_pca (37421, 30)
```

```
#%%
# Scree Plot
y = pca.explained_variance_ratio_

x = np.array([i for i in range(1, len(y)+1)])

plt.plot(x, y, 'r-x')
```

| Model | Accuracy | recall | Specificity | AUC Sc |
|---|---|---|---|---|
| Logistic Regression | 64.60% | 94.32% | 25.25% | 0.5978 |
| Nave Bayes | 61.74% | 89.47% | 25.02% | 0.5724 |
| Random Forest | 93.54% | 97% | 90.08% | 0.9354 |

| | | | | |
|---|---|---|---|---|
| Linear-SVC | 64.55% | 94.85% | 24.44% | 0.5958 |
| Nu-SVC | 78.63% | 92.36% | 60.45% | 0.7640 |
| C-SVC | 67.55% | 98.13% | 27.06% | 0.626 |
| Neural Network1 | 65.01% | 94.31% | 26.21% | 0.6026 |
| Neural Network2 | 65.96% | 97.02% | 24.83% | 0.6092 |

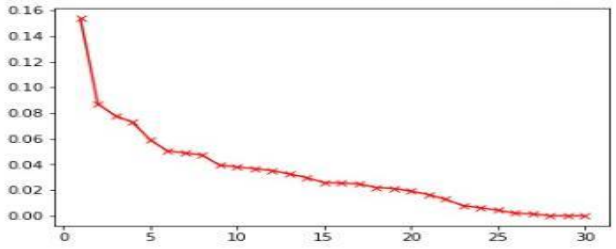
Performances of algorithm on PCA dataset

```
# Scree Plot
y = pca.explained_variance_ratio_

x = np.array([i for i in range(1, len(y)+1)])

plt.plot(x, y, 'r-x')
```

```
[<matplotlib.lines.Line2D at 0x7f68bfb57c18>]
```




```
# Filter principle components
sum(pca.explained_variance_ratio_[:14])
X_pca = X_pca[:,:14] # Slice the first 14 features
X_pca.shape
```

```
(37421, 14)
```

## C - Support Vector Regression

```
from sklearn.svm import SVC
```

```
svc = SVC(gamma='auto')
svcgridsearch = GridSearchCV(svc, cv = skf, param_grid = params, return_train_score = True)
svcgridsearch.fit(x_train, y_train)
```

```
GridSearchCV(cv=StratifiedKFold(n_splits=10, random_state=None, shuffle=False),
             error_score=nan,
             estimator=SVC(C=1.0, break_ties=False, cache_size=200,
                           class_weight=None, coef0=0.0,
                           decision_function_shape='ovr', degree=3,
                           gamma='auto', kernel='rbf', max_iter=-1,
                           probability=False, random_state=None, shrinking=True,
                           tol=0.001, verbose=False),
             iid='deprecated', n_jobs=None, param_grid={},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=True,
             scoring=None, verbose=0)
```

```
svcgridsearch.cv_results_
svcgridsearch.score(X_pca, new_df.IsAlert)
svc_predict = svcgridsearch.predict(X_pca)
```

## Neural Network

```
[ ]  from sklearn.neural_network import MLPClassifier
```

```
[ ]  nngridsearch.cv_results_
     nngridsearch.score(X_pca, new_df.IsAlert)
     nn_predict = nngridsearch.predict(X_pca)
```

```
[ ]  confusion_matrix(new_df.IsAlert, nn_predict)
```

```
array([[ 4205, 11897],
       [ 1205, 20114]])
```

```
roc_auc_score(new_df.IsAlert, nn_predict)
```

```
0.6023126662815139
```

## OUTPUT SCREENS

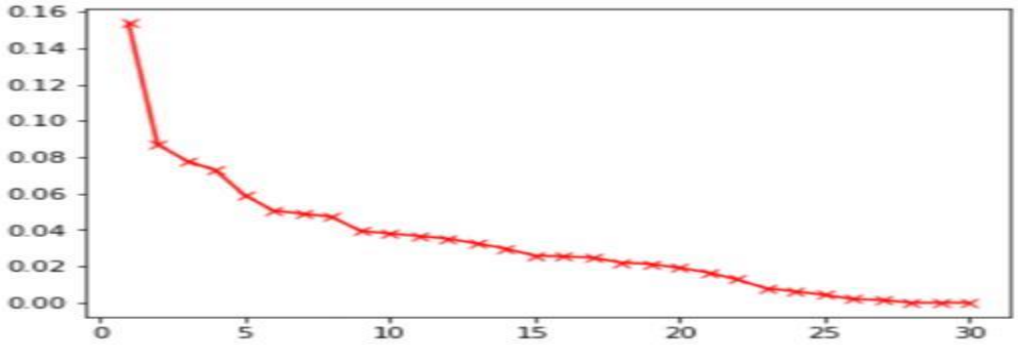



**FIG 1: Scree Plot of 30 features**

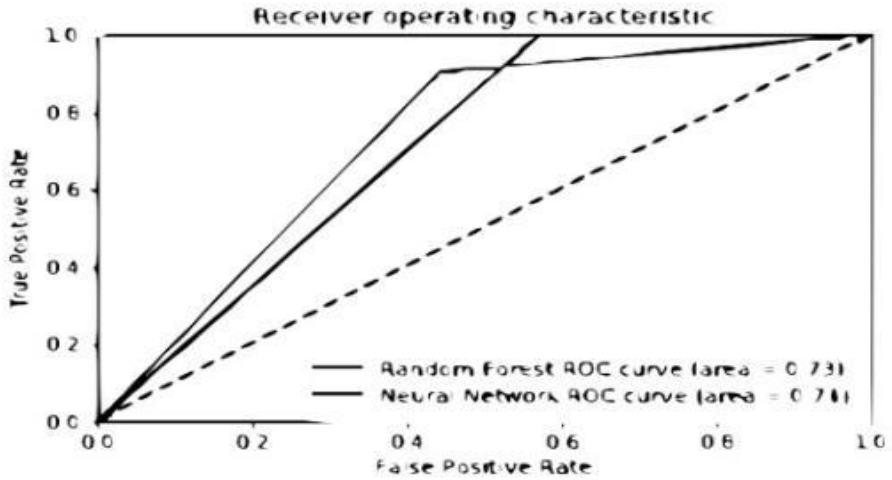



**FIG 2: ROC curve of two models**

# REFERENCES


- **https://www.kaggle.com/c/stayalert/data**

## GITHUB LINK

**https://github.com/bindhu520/Safe-driving-Challenge-ML-PROECT-**