DRIVER ALERT DETECTION SYSTEM

Report of EC-400M Mid_term Project Presentation

By

SWET SHEERSH(BTECH/60309/19)

SEJAL KUMARI(BTECH/60020/19)

Under the supervision of

Dr. Rajesh Kumar Lal

Dept. of Electronics Comm. Engg.
BIT Mesra, Deoghar Campus

Dept. of Electronics & Communication Engg.
Birla Institute of Technology, Mesra, Deoghar Campus
November 2022

CONTENTS

Overview :

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 objective of this Project is to design a detector/classifier that will detect whether the driver is alert or not alert, employing any combination of vehicular, environmental and driver physiological data that are acquired while driving.

Common distracted driving behaviors:

Distracted driving comes in a variety of forms. These behaviors include anything that inhibits a driver from paying full at ention to the task of driving, or inhibits them from being fully engaged to adequately respond to changes in the driving environment.

- Drowsy driving
- Using smart phone while Driving
- Texting
- Eating
- Smoking
- Using a tablet
- Reading paperwork
- Working on infotainment system

UNIQUENESS OF OUR MODAL:

To reduce injuries, to themselves or others, and avoid collisions. In order to achieve this, they need a driver alert system that:

- Alerts drivers before collisions happen; that is, avoid collisions rather than report on the collision after the fact.
- Helps drivers improve their driving; especially newer drivers who are statistically several times more likely to be involved in collisions
- Our System Environment will intimate the Driver as well as Owner of vechile by using Alarm System & sending a Message on Smartphone.

Once we got the Data ,we can not implement ML algorithm(Model) Directly.As we allready know ,for Model implementation Data set should be in specific formet(Numerical value with NO NULL and Error). To achieve it, we used unsupervised techniques which are following:

- ➢ Imputing missing numeric values
- ➢ Encoding Categorical Data
- ➢ Scaling Numeric Features
- ➢ Training, Validation and Test Sets
- ➢ Feature selection and Reduction Techniques

Once we get preprocessed Data ,Now we can Train Model and Hyperparameter Tuning for achieving better accuracy on test Data set.

✓ **Finally ,I selected Random Forest and achieved 99% accuracy on validation Data set and 83% on test data set.**

Driver Alertness Detection -- - Machine Learning project

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 objective of this challenge is to design a detector/classifier that will detect whether the driver is alert or not alert, employing any combination of vehicular, environmental and driver physiological data that are acquired while driving.

```
!pip install jovian --upgrade --quiet
!pip install xgboost --upgrade --quiet
```

```
     |████████████████████████████████| 173.5 MB 11 kB/s
```

```
import jovian
```

```
# Execute this to save new versions of the notebook
jovian.commit(project="Driver Alertness Detection")
```

```
[jovian] Detected Colab notebook...
[jovian] Please enter your API key ( from https://jovian.ai/ ):
API KEY: ··········
[jovian] Uploading colab notebook to Jovian...
Committed successfully! https://jovian.ai/btech60309-19/driver-alertness-detection
```

```
'https://jovian.ai/btech60309-19/driver-alertness-detection'
```

Downloading the dataset

```
!pip install opendatasets --upgrade
import opendatasets as od
```

```
Collecting opendatasets
  Downloading opendatasets-0.1.20-py3-none-any.whl (14 kB)
Requirement already satisfied: kaggle in /usr/local/lib/python3.7/dist-packages (from
opendatasets) (1.5.12)
Requirement already satisfied: tqdm in /usr/local/lib/python3.7/dist-packages (from
opendatasets) (4.62.3)
Requirement already satisfied: click in /usr/local/lib/python3.7/dist-packages (from
opendatasets) (7.1.2)
Requirement already satisfied: python-slugify in /usr/local/lib/python3.7/dist-packages
(from kaggle->opendatasets) (5.0.2)
Requirement already satisfied: certifi in /usr/local/lib/python3.7/dist-packages (from
kaggle->opendatasets) (2021.10.8)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages (from
kaggle->opendatasets) (2.23.0)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.7/dist-packages (from
kaggle->opendatasets) (1.24.3)
Requirement already satisfied: six>=1.10 in /usr/local/lib/python3.7/dist-packages
(from kaggle->opendatasets) (1.15.0)
Requirement already satisfied: python-dateutil in /usr/local/lib/python3.7/dist-
packages (from kaggle->opendatasets) (2.8.2)
Requirement already satisfied: text-unidecode>=1.3 in /usr/local/lib/python3.7/dist-
packages (from python-slugify->kaggle->opendatasets) (1.3)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages
(from requests->kaggle->opendatasets) (2.10)
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.7/dist-
packages (from requests->kaggle->opendatasets) (3.0.4)
Installing collected packages: opendatasets
Successfully installed opendatasets-0.1.20
```

```
od.download('https://www.kaggle.com/c/stayalert/data?select=fordTrain.csv')
```

```
Please provide your Kaggle credentials to download this dataset. Learn more:
http://bit.ly/kaggle-creds
Your Kaggle username: swetsheersh
Your Kaggle Key: ··········
Downloading stayalert.zip to ./stayalert

100%|██████████| 18.2M/18.2M [00:00<00:00, 67.4MB/s]


Extracting archive ./stayalert/stayalert.zip to ./stayalert
```

```
import os
```

```
os.listdir('./stayalert')
```

```
['example_submission.csv', 'fordTrain.csv', 'Solution.csv', 'fordTest.csv']
```

```
import pandas as pd
```

```
fordtest=pd.read_csv('./stayalert/fordTest.csv')
```

```
submission=pd.read_csv('./stayalert/example_submission.csv')
```

```
fordtrain=pd.read_csv('./stayalert/fordTrain.csv')
```

```
solution=pd.read_csv('./stayalert/Solution.csv')
```

Problem Statement

Driving while not alert can be deadly. The objective is to design a classifier that will detect whether the driver is alert or not alert, employing data that are acquired while driving.

riving 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 objective of this challenge is to design a detector/classifier that will detect whether the driver is alert or not alert, employing any combination of vehicular, environmental and driver physiological data that are acquired while driving.

The data for this challenge shows the results of a number of "trials", each one representing about 2 minutes of sequential data that are recorded every 100 ms 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;

The third column values are hidden in the test set ('fordTest.csv').

The file 'example_submission.csv' is an example of a submission file - your submission files should be in exactly the same format, with only values in the last column ('Prediction') different. Predictions are expected to be real numbers between 0 and 1 inclusive.

Note: The actual names and measurement units of the physiological, environmental and vehicular data are not disclosed in this challenge. Models which use fewer physiological variables (columns with names starting with 'P') are of particular interest, therefore competitors are encouraged to consider models which require fewer of these variables.

```
fordtest
```

	TrialID	ObsNum	IsAlert	P1	P2	P3	P4	P5	P6	P7	P8	E1	E
0	0	0	?	38.4294	10.94350	1000	60.0000	0.302277	508	118.1100	0	0.000	0.00
1	0	1	?	38.3609	15.32120	1000	60.0000	0.302277	508	118.1100	0	0.000	0.00
2	0	2	?	38.2342	11.51400	1000	60.0000	0.302277	508	118.1100	0	0.000	0.00
3	0	3	?	37.9304	12.26150	1000	60.0000	0.302277	508	118.1100	0	0.000	0.00
4	0	4	?	37.8085	12.36660	1000	60.0000	0.302277	504	119.0480	0	0.000	0.00
...	...	...	...	...	...	...	...	...	...	...	...	...	
120835	99	1206	?	37.3798	17.40260	892	67.2646	0.131030	752	79.7872	0	0.068	214.03
120836	99	1207	?	37.1653	5.37419	892	67.2646	0.131030	752	79.7872	0	0.068	214.03
120837	99	1208	?	36.9131	9.26657	892	67.2646	0.131030	752	79.7872	0	0.068	214.03
120838	99	1209	?	36.6297	10.41710	892	67.2646	0.131030	752	79.7872	0	0.068	214.03
120839	99	1210	?	36.6297	10.41710	892	67.2646	0.131030	752	79.7872	0	0.068	214.03

120840 rows × 33 columns

```
fordtrain
```

	TrialID	ObsNum	IsAlert	P1	P2	P3	P4	P5	P6	P7	P8	E1	E
0	0	0	0	34.7406	9.84593	1400	42.8571	0.290601	572	104.8950	0	0.000	0.0
1	0	1	0	34.4215	13.41120	1400	42.8571	0.290601	572	104.8950	0	0.000	0.0
2	0	2	0	34.3447	15.18520	1400	42.8571	0.290601	576	104.1670	0	0.000	0.0
3	0	3	0	34.3421	8.84696	1400	42.8571	0.290601	576	104.1670	0	0.000	0.0
4	0	4	0	34.3322	14.69940	1400	42.8571	0.290601	576	104.1670	0	0.000	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	
604324	510	1194	1	32.0051	10.13240	800	75.0000	0.081731	680	88.2353	0	17.807	222.1

	TrialID	ObsNum	IsAlert	P1	P2	P3	P4	P5	P6	P7	P8	E1	E
604325	510	1195	1	32.0393	12.45040	800	75.0000	0.081731	680	88.2353	0	17.807	222.1
604326	510	1196	1	32.0762	10.06180	800	75.0000	0.081731	680	88.2353	0	17.807	222.1
604327	510	1197	1	32.1154	17.84500	800	75.0000	0.081731	680	88.2353	0	17.807	222.1
604328	510	1198	1	32.1154	17.84500	800	75.0000	0.081731	680	88.2353	0	17.807	222.1

604329 rows × 33 columns

```
submission
```

	TrialID	ObsNum	Prediction
0	0	0	0
1	0	1	0
2	0	2	0
3	0	3	0
4	0	4	0
...	...	...	...
120835	99	1206	0
120836	99	1207	0
120837	99	1208	0
120838	99	1209	0
120839	99	1210	0

120840 rows × 3 columns

```
fordtrain.ObsNum.nunique()
```

```
1211
```

```
fordtrain.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 604329 entries, 0 to 604328
Data columns (total 33 columns):
 #   Column   Non-Null Count   Dtype  
---  ------   --------------   -----  
 0   TrialID  604329 non-null  int64  
 1   ObsNum   604329 non-null  int64  
 2   IsAlert  604329 non-null  int64  
 3   P1       604329 non-null  float64
 4   P2       604329 non-null  float64
 5   P3       604329 non-null  int64  
 6   P4       604329 non-null  float64
 7   P5       604329 non-null  float64
 8   P6        604329 non-null  int64
 9   P7        604329 non-null  float64
 10  P8        604329 non-null  int64
 11  E1        604329 non-null  float64
 12  E2        604329 non-null  float64
 13  E3        604329 non-null  int64
 14  E4        604329 non-null  int64
 15  E5        604329 non-null  float64
 16  E6        604329 non-null  int64
 17  E7        604329 non-null  int64
 18  E8        604329 non-null  int64
 19  E9        604329 non-null  int64
 20  E10       604329 non-null  int64
 21  E11       604329 non-null  float64
 22  V1        604329 non-null  float64
 23  V2        604329 non-null  float64
 24  V3        604329 non-null  int64
 25  V4        604329 non-null  float64
 26  V5        604329 non-null  int64
 27  V6        604329 non-null  int64
 28  V7        604329 non-null  int64
 29  V8        604329 non-null  float64
 30  V9        604329 non-null  int64
 31  V10       604329 non-null  int64
 32  V11       604329 non-null  float64
dtypes: float64(14), int64(19)
memory usage: 152.2 MB
```

```
fordtrain.corr()
```

	TrialID	ObsNum	IsAlert	P1	P2	P3	P4	P5	P6	
TrialID	1.000000	-0.000162	-0.145816	0.016772	-0.004473	0.000369	0.001880	0.022632	0.005377	0.11
ObsNum	-0.000162	1.000000	-0.005143	0.018324	-0.001764	0.002199	-0.001191	0.005568	-0.015791	0.00
IsAlert	-0.145816	-0.005143	1.000000	0.018361	0.014383	0.005168	-0.008177	0.038160	-0.000400	0.18
P1	0.016772	0.018324	0.018361	1.000000	-0.006674	-0.010317	0.011704	0.010911	0.045429	0.02
P2	-0.004473	-0.001764	0.014383	-0.006674	1.000000	-0.002539	0.002132	0.008390	-0.022003	0.05
P3	0.000369	0.002199	0.005168	-0.010317	-0.002539	1.000000	-0.944435	0.035129	0.012444	-0.00
P4	0.001880	-0.001191	-0.008177	0.011704	0.002132	-0.944435	1.000000	-0.032897	-0.010627	0.00
P5	0.022632	0.005568	0.038160	0.010911	0.008390	0.035129	-0.032897	1.000000	0.002314	-0.02
P6	0.005377	-0.015791	-0.000400	0.045429	-0.022003	0.012444	-0.010627	0.002314	1.000000	-0.12
P7	0.111903	0.003498	0.189796	0.027461	0.052171	-0.006097	0.007323	-0.023628	-0.125580	1.00
P8	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

	TrialID	ObsNum	IsAlert	P1	P2	P3	P4	P5	P6	
E1	-0.061881	-0.000122	-0.160830	-0.015436	-0.012045	0.005795	-0.004870	-0.062955	-0.006273	-0.08
E2	0.015610	-0.003558	-0.105495	-0.009356	-0.019121	0.013007	-0.010749	-0.033420	0.006025	-0.09
E3	-0.087071	0.002931	0.157973	0.024131	0.062076	-0.016437	0.010696	0.069444	-0.025157	0.29
E4	-0.050151	-0.004580	0.047992	-0.010574	0.003529	0.000515	0.001580	-0.002757	-0.001288	0.02
E5	0.234524	0.010314	-0.067453	-0.006564	-0.005140	0.004496	-0.003086	-0.020218	-0.007514	0.05
E6	-0.034418	-0.003838	-0.189198	-0.004635	-0.006843	-0.004508	0.004230	-0.006387	0.011529	-0.10
E7	-0.116919	0.002005	-0.329722	-0.013194	-0.002058	-0.014086	0.013755	-0.032576	-0.006923	-0.04
E8	-0.095434	0.009400	-0.283440	-0.010918	0.002920	-0.014135	0.012959	-0.048551	-0.008941	0.02
E9	0.116988	-0.002779	0.380353	0.004688	0.002266	0.018113	-0.016615	0.005177	-0.005549	0.04
E10	-0.100521	-0.008684	-0.067051	0.004549	0.014589	-0.013289	0.010501	-0.074753	-0.034470	0.10
E11	0.073676	0.004978	0.079002	0.015882	0.001857	0.007423	-0.005681	0.028216	-0.008939	0.08
V1	-0.117728	0.004242	-0.269967	-0.025763	0.011310	-0.011347	0.010061	-0.054428	-0.024449	-0.04
V2	0.065063	0.019009	-0.050740	-0.021118	0.001779	0.008651	-0.006926	0.026232	-0.010418	0.02
V3	0.001900	0.007753	-0.062000	0.002551	0.002272	-0.006380	0.008049	0.005371	-0.004996	0.00
V4	-0.074320	-0.000480	0.097022	0.021404	-0.006038	0.013045	-0.010905	0.070290	0.019121	0.03
V5	0.123721	-0.006284	0.055429	0.051348	-0.023902	0.001312	0.001705	-0.016671	0.029222	0.02
V6	-0.097389	0.003935	-0.244150	-0.019792	0.010608	-0.010245	0.008348	-0.046353	-0.025728	-0.02
V7	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
V8	-0.047593	0.008191	-0.165550	-0.029747	0.008257	-0.007963	0.005379	-0.023359	-0.016850	-0.00
V9	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
V10	-0.093818	0.005145	-0.259607	-0.004563	0.001946	-0.009630	0.008906	-0.022193	-0.002163	-0.04
V11	0.078887	-0.011465	0.155722	0.344636	-0.034248	-0.009808	0.009841	-0.004897	0.012783	0.01

```
fordtest.isna().sum()
```

```
TrialID    0
ObsNum     0
IsAlert    0
P1         0
P2         0
P3         0
P4         0
P5         0
P6         0
P7         0
P8         0
E1         0
E2         0
E3         0
E4         0
E5         0
E6         0
E7         0
```

```
E8         0
E9         0
E10        0
E11        0
V1         0
V2         0
V3         0
V4         0
V5         0
V6         0
V7         0
V8         0
V9         0
V10        0
V11        0
dtype: int64
```

```python
fordtest.describe()
```

	TrialID	ObsNum	P1	P2	P3	P4	
count	120840.000000	120840.000000	120840.000000	120840.000000	120840.000000	120840.000000	120840.0
mean	49.496491	603.711635	35.450222	12.008451	1026.668355	64.148812	0.1
std	28.865733	348.856410	3.303869	4.351161	310.874514	19.995102	0.1
min	0.000000	0.000000	17.776300	-25.911800	504.000000	25.996500	0.0
25%	24.000000	302.000000	33.456300	9.600658	788.000000	49.180300	0.0
50%	49.000000	604.000000	34.877800	11.288900	1000.000000	60.000000	0.1
75%	74.000000	906.000000	36.862200	13.542925	1220.000000	76.142100	0.1
max	99.000000	1210.000000	81.819600	39.757300	2308.000000	119.048000	4.6

```
!pip install plotly
```

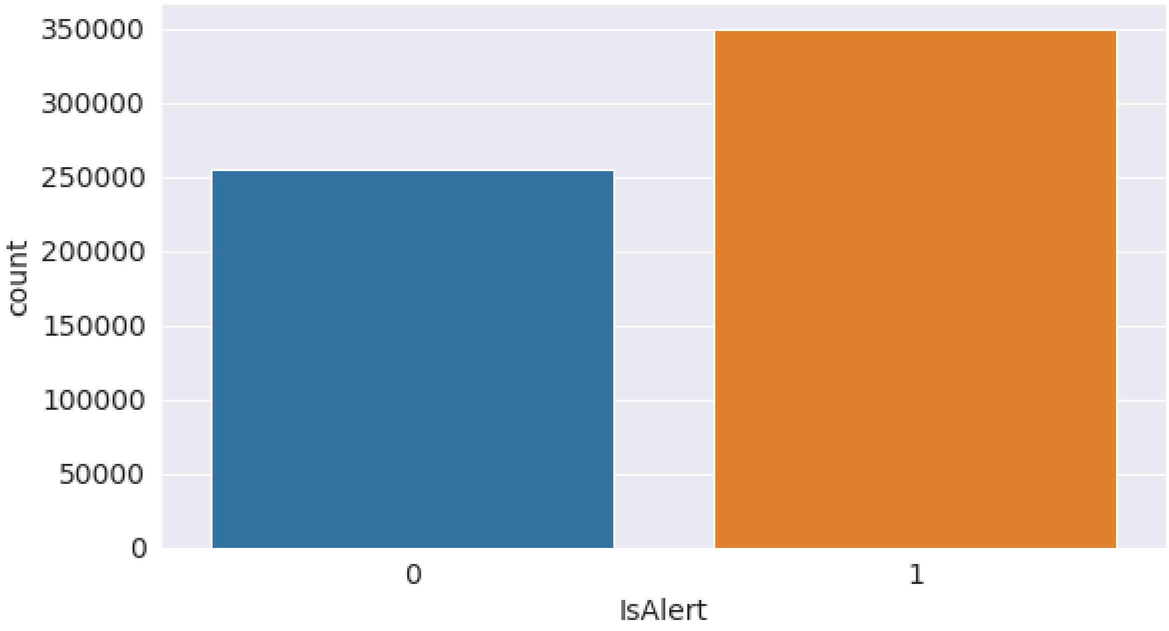
```
Requirement already satisfied: plotly in /usr/local/lib/python3.7/dist-packages (4.4.1)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from plotly) (1.15.0)
Requirement already satisfied: retrying>=1.3.3 in /usr/local/lib/python3.7/dist-packages (from plotly) (1.3.3)
```

```python
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
%matplotlib inline

sns.set_style("darkgrid")
matplotlib.rcParams['font.size'] = 14
matplotlib.rcParams['figure.figsize'] = (9, 5)
matplotlib.rcParams['figure.facecolor'] = '#00000000'
```

```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', 150)
```

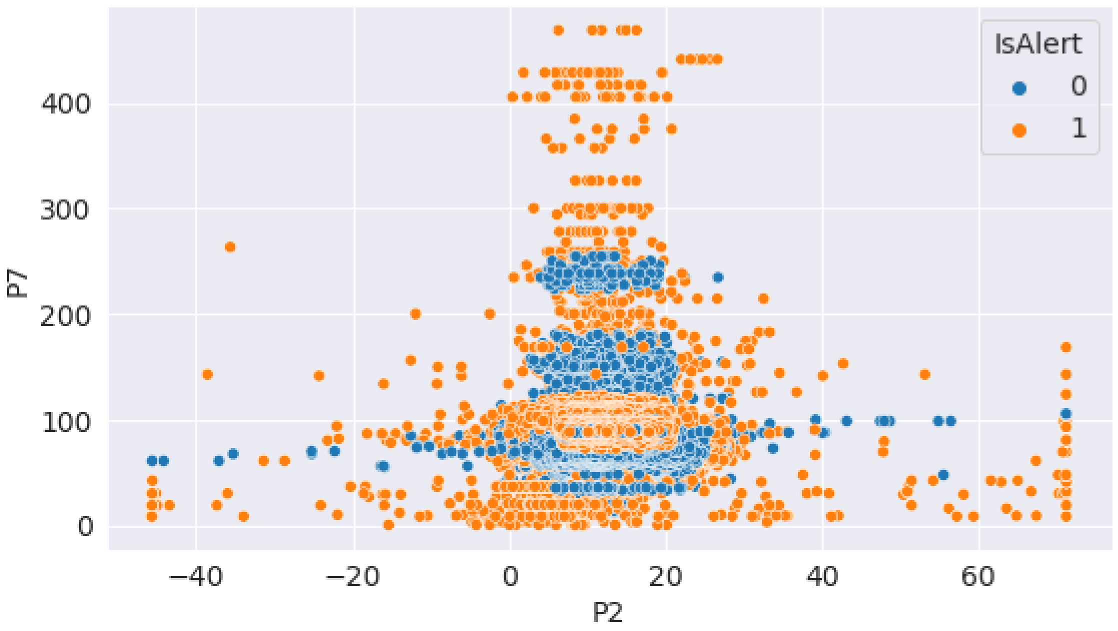
```
sns.countplot(fordtrain.IsAlert)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variable as a keyword arg: x. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.


<matplotlib.axes._subplots.AxesSubplot at 0x7f3dfb1423d0>
```



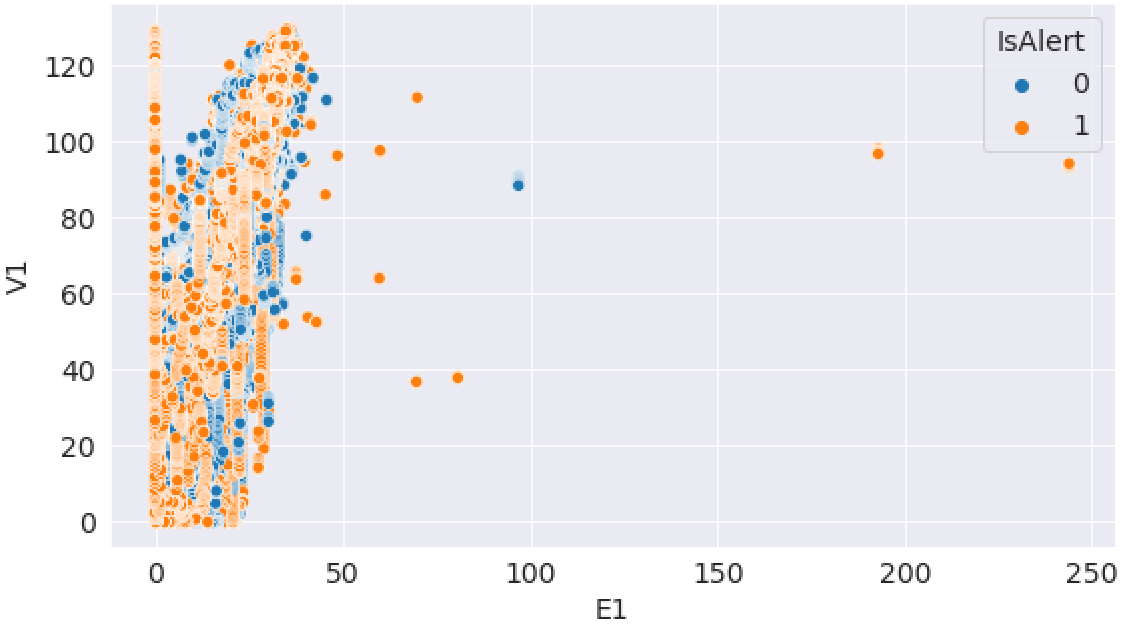
```
sns.scatterplot(fordtrain.P2,fordtrain.P7,hue=fordtrain.IsAlert)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.


<matplotlib.axes._subplots.AxesSubplot at 0x7f3dfb08c310>
```



```
sns.scatterplot(fordtrain.E1,fordtrain.V1,hue=fordtrain.IsAlert)
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning:

Pass the following variables as keyword args: x, y. From version 0.12, the only valid
positional argument will be `data`, and passing other arguments without an explicit
keyword will result in an error or misinterpretation.


<matplotlib.axes._subplots.AxesSubplot at 0x7f3df98f4a90>
```



```
fordtrain.ObsNum.count()
```

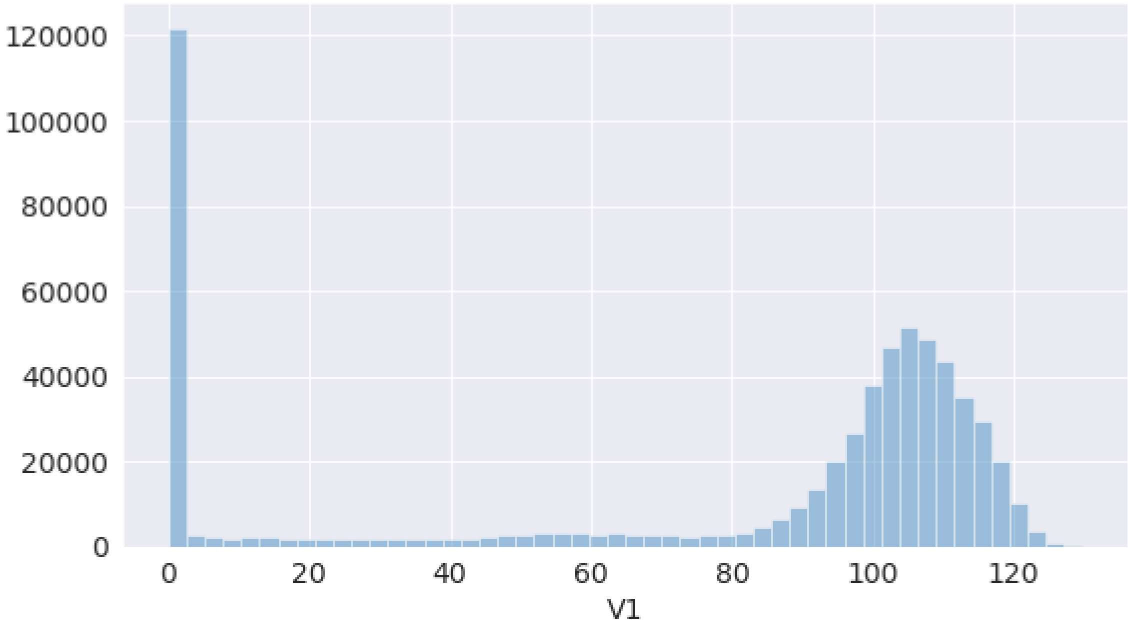
```
604329
```

```
px.scatter(fordtrain,x='E1',y='V1',color='IsAlert')
```

Output hidden; open in https://colab.research.google.com to view.

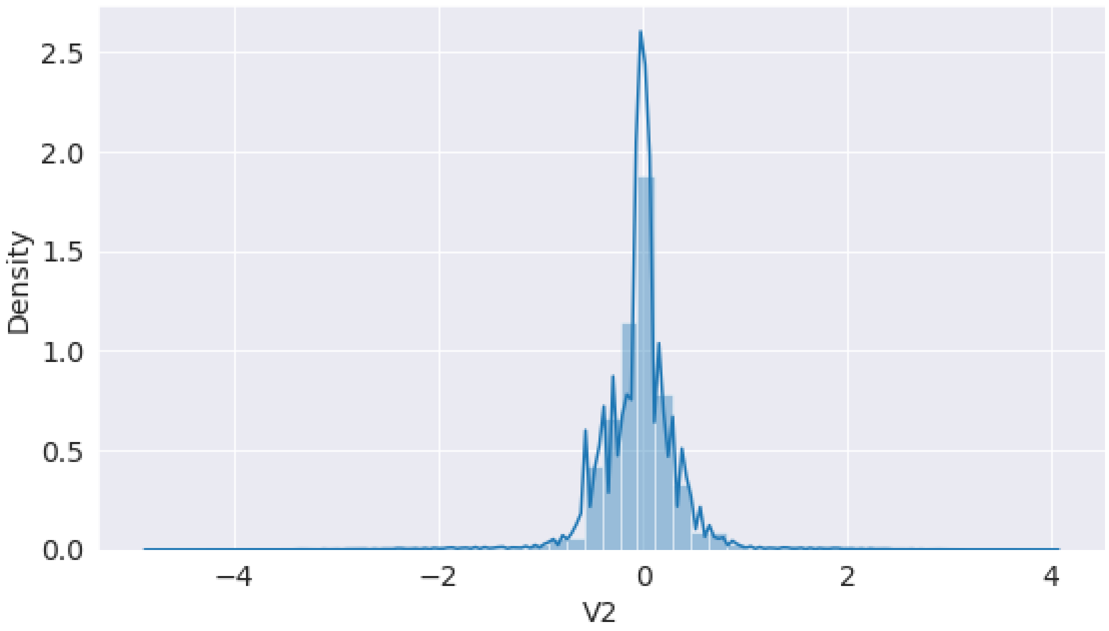
```
sns.distplot(fordtrain.V1, kde=False);
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
```



```
sns.distplot(fordtrain.V2);
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/distributions.py:2619: FutureWarning:

`distplot` is a deprecated function and will be removed in a future version. Please
adapt your code to use either `displot` (a figure-level function with similar
flexibility) or `histplot` (an axes-level function for histograms).
```



Preparing the Data for Training

```
fordtrain.columns
```

```
Index(['TrialID', 'ObsNum', 'IsAlert', 'P1', 'P2', 'P3', 'P4', 'P5', 'P6',
       'P7', 'P8', 'E1', 'E2', 'E3', 'E4', 'E5', 'E6', 'E7', 'E8', 'E9', 'E10',
       'E11', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11'],
      dtype='object')
```

```
input=['P1', 'P2', 'P3', 'P4', 'P5', 'P6',
       'P7', 'P8', 'E1', 'E2', 'E3', 'E4', 'E5', 'E6', 'E7', 'E8', 'E9', 'E10',
       'E11', 'V1', 'V2', 'V3', 'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10',
       'V11']
output='IsAlert'
cat_cols=['TrialID', 'ObsNum']
```

```
x_train=fordtrain[input + cat_cols]
train_target=fordtrain[output]
x_test=fordtest[input +cat_cols]
test_target=fordtest[output]
```

```
x_train
```

	P1	P2	P3	P4	P5	P6	P7	P8	E1	E2	E3	E4	E5	E
0	34.7406	9.84593	1400	42.8571	0.290601	572	104.8950	0	0.000	0.00	1	-20	0.015875	32
1	34.4215	13.41120	1400	42.8571	0.290601	572	104.8950	0	0.000	0.00	1	-20	0.015875	32
2	34.3447	15.18520	1400	42.8571	0.290601	576	104.1670	0	0.000	0.00	1	-20	0.015875	32
3	34.3421	8.84696	1400	42.8571	0.290601	576	104.1670	0	0.000	0.00	1	-20	0.015875	32
4	34.3322	14.69940	1400	42.8571	0.290601	576	104.1670	0	0.000	0.00	1	-20	0.015875	32

	P1	P2	P3	P4	P5	P6	P7	P8	E1	E2	E3	E4	E5	E
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
604324	32.0051	10.13240	800	75.0000	0.081731	680	88.2353	0	17.807	222.11	0	0	0.016379	32
604325	32.0393	12.45040	800	75.0000	0.081731	680	88.2353	0	17.807	222.11	0	0	0.016379	32
604326	32.0762	10.06180	800	75.0000	0.081731	680	88.2353	0	17.807	222.11	0	0	0.016379	32
604327	32.1154	17.84500	800	75.0000	0.081731	680	88.2353	0	17.807	222.11	0	0	0.016379	32
604328	32.1154	17.84500	800	75.0000	0.081731	680	88.2353	0	17.807	222.11	0	0	0.016379	32

604329 rows × 32 columns

Imputing missing numeric values

```
from sklearn.impute import SimpleImputer
```

```
imputer=SimpleImputer(strategy='mean')
```

```
imputer.fit(x_train[input])
```

```
SimpleImputer()
```

```
x_train[input]=imputer.transform(x_train[input])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
x_test[input]=imputer.transform(x_test[input])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

Encoding Categorical Data

```
from sklearn.preprocessing import OneHotEncoder
```

```
encoder=OneHotEncoder(sparse=False, handle_unknown='ignore').fit(x_train[cat_cols])
```

```
encoded_cols=list(encoder.get_feature_names(cat_cols))
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning:

Function get_feature_names is deprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out instead.
```

```
len(encoded_cols)
```

```
1711
```

```
#x_train[encoded_cols]=encoder.transform(x_train[cat_cols])
#x_test[encoded_cols]=encoder.transform(x_test[cat_cols])
```

```
jovian.commit()
```

```
[jovian] Detected Colab notebook...
```

```
[jovian] Uploading colab notebook to Jovian...
Committed successfully! https://jovian.ai/btech60309-19/driver-alertness-detection
```

```
'https://jovian.ai/btech60309-19/driver-alertness-detection'
```

Scaling Numeric Features

```python
from sklearn.preprocessing import MinMaxScaler
```

```python
scaler=MinMaxScaler().fit(x_train[input])
```

```python
x_train[input].describe().loc[['min', 'max']]
```

	P1	P2	P3	P4	P5	P6	P7	P8	E1	E2	E3	E4
min	-22.4812	-45.6292	504.0	23.8853	0.03892	128.0	0.262224	0.0	0.000	0.000	0.0	-250.0
max	101.3510	71.1737	2512.0	119.0480	27.20220	228812.0	468.750000	0.0	243.991	359.995	4.0	260.0

```python
x_train[input]=scaler.transform(x_train[input])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```python
x_test[input]=scaler.transform(x_test[input])
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:1: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy

/usr/local/lib/python3.7/dist-packages/pandas/core/indexing.py:1734:
SettingWithCopyWarning:


A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
x_train=x_train[input ]
```

```
x_test=x_test[input]
```

```
jovian.commit()
```

```
[jovian] Detected Colab notebook...
[jovian] Uploading colab notebook to Jovian...
Committed successfully! https://jovian.ai/btech60309-19/driver-alertness-detection
```

```
'https://jovian.ai/btech60309-19/driver-alertness-detection'
```

Training, Validation and Test Sets

```
from sklearn.model_selection import train_test_split
```

```
train_df,val_df,train_target,val_target = train_test_split(x_train,train_target, test_s
```

```
jovian.commit()
```

```
[jovian] Detected Colab notebook...
[jovian] Uploading colab notebook to Jovian...
Committed successfully! https://jovian.ai/btech60309-19/driver-alertness-detection
```

```
'https://jovian.ai/btech60309-19/driver-alertness-detection'
```

```
jovian.commit()
```

```
[jovian] Detected Colab notebook...
[jovian] Uploading colab notebook to Jovian...
Committed successfully! https://jovian.ai/btech60309-19/driver-alertness-detection
```

```
'https://jovian.ai/btech60309-19/driver-alertness-detection'
```

Training LogisticRegression

```python
from sklearn.linear_model import LogisticRegression
```

```python
model = LogisticRegression(solver='liblinear')
```

```python
model.fit(train_df,train_target)
```

```
LogisticRegression(solver='liblinear')
```

```python
print(model.coef_.tolist())
```

```
[[-1.7891672811875508, -0.35266232993615465, -0.3779486650421321, -0.2830247685033119,
5.655148515900745, 7.496776978845858, 5.048188327673843, 0.0, 1.2481520947286018,
-0.3074197610756305, 0.5415739996669995, -0.018389553942291186, -1.3471881880639474,
-3.788474107176254, -1.0645956444350084, -1.971924966940731, 3.180573415589189,
1.0265922966193521, -0.12430131580524675, -1.8971786500756127, -0.6059740609088667,
-0.31748754660722406, -0.21274063895391376, -0.08765310993027522, 1.8517687137092007,
0.0, -0.5361062696905858, 0.0, -0.7576070322022002, 48.542501605569626]]
```

```python
print(model.intercept_)
```

```
[-0.54542233]
```

```python
train_preds = model.predict(train_df)
```

```python
train_probs = model.predict_proba(train_df)
train_probs
```

```
array([[0.09298905, 0.90701095],
       [0.46530983, 0.53469017],
       [0.28690164, 0.71309836],
       ...,
       [0.54165955, 0.45834045],
       [0.49910443, 0.50089557],
       [0.03751907, 0.96248093]])
```

```python
from sklearn.metrics import accuracy_score,confusion_matrix
```

```
accuracy_score(train_target, train_preds)
```

```
0.7915083470710271
```

```
accuracy_score(val_target, model.predict(val_df))
```

```
0.790395975708636
```

```
confusion_matrix(train_preds, train_target, normalize='true')
```

```
array([[0.78012866, 0.21987134],
       [0.2015345 , 0.7984655 ]])
```

```
model.feature_names_in_
```

```
array(['P1', 'P2', 'P3', 'P4', 'P5', 'P6', 'P7', 'P8', 'E1', 'E2', 'E3',
       'E4', 'E5', 'E6', 'E7', 'E8', 'E9', 'E10', 'E11', 'V1', 'V2', 'V3',
       'V4', 'V5', 'V6', 'V7', 'V8', 'V9', 'V10', 'V11'], dtype=object)
```

From Logistic Regression I got Accuracy of 79% on validation data set

DecisionTree

```
from sklearn.tree import DecisionTreeClassifier
```

```
tree=DecisionTreeClassifier(random_state=42)
```

```
tree.fit(train_df, train_target)
```

```
DecisionTreeClassifier(random_state=42)
```

```
train_preds1 = tree.predict(train_df)
```

```
accuracy_score(train_target,train_preds1)
```

```
1.0
```

```
accuracy_score(val_target, tree.predict(val_df))
```

```
0.9877302136250062
```

```
tree.max_features_
```

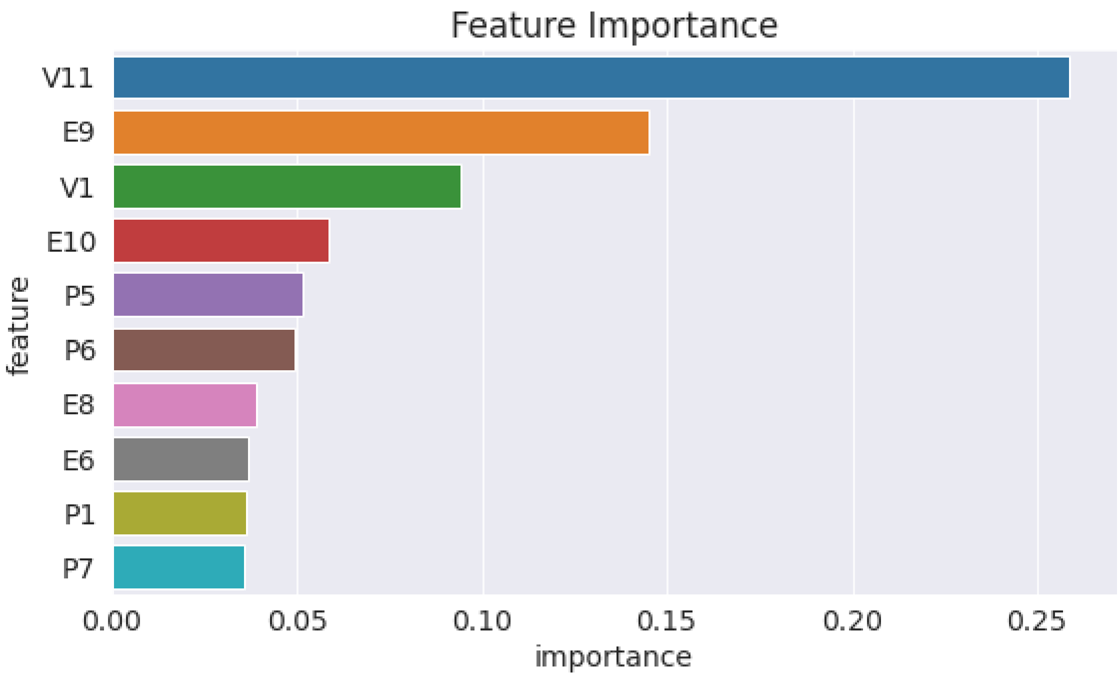
```
30
```

```
tree.feature_importances_
```

```
array([0.0365226 , 0.00157197, 0.00528065, 0.00568358, 0.05182447,
       0.04953578, 0.03613309, 0.        , 0.01099683, 0.02597018,
       0.00039367, 0.01646381, 0.02989241, 0.03724177, 0.02379401,
       0.0389831 , 0.14488117, 0.05841389, 0.00260946, 0.09449234,
       0.01003286, 0.00214564, 0.01381794, 0.00495119, 0.02817809,
       0.        , 0.01017418, 0.        , 0.00145306, 0.25856226])
```

```
importance_df = pd.DataFrame({
    'feature':train_df.columns,
    'importance': tree.feature_importances_
}).sort_values('importance', ascending=False)
```

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.title('Feature Importance')
sns.barplot(data=importance_df.head(10), x='importance', y='feature');
```



```
def max_depth_error(md):
    model = DecisionTreeClassifier(max_depth=md, random_state=42)
    model.fit(train_df, train_target)
    train_acc = 1 - model.score(train_df, train_target)
    val_acc = 1 - model.score(val_df, val_target)
    return {'Max Depth': md, 'Training Error': train_acc, 'Validation Error': val_acc}
```

```
%%time
errors_df = pd.DataFrame([max_depth_error(md) for md in range(1, 21)])
```

```
CPU times: user 2min 28s, sys: 260 µs, total: 2min 28s
Wall time: 2min 27s
```

```
errors_df
```

	Max Depth	Training Error	Validation Error
0	1	0.316949	0.318038
1	2	0.237507	0.236791
2	3	0.203848	0.203680
3	4	0.187603	0.189416
4	5	0.149587	0.150464
5	6	0.134500	0.135034
6	7	0.119167	0.119893
7	8	0.107408	0.108591
8	9	0.094160	0.095263
9	10	0.081456	0.082778
10	11	0.070903	0.072742
11	12	0.062365	0.064882
12	13	0.053866	0.057684
13	14	0.045954	0.050461
14	15	0.040268	0.045455
15	16	0.034940	0.040301
16	17	0.030586	0.036933
17	18	0.026898	0.033972
18	19	0.023975	0.031340
19	20	0.021267	0.029206

```
tree=DecisionTreeClassifier(random_state=42,max_depth=30)
```

```
tree.fit(train_df, train_target)
```

```
DecisionTreeClassifier(max_depth=30, random_state=42)
```

```
tree.score(train_df, train_target)
```

```
0.9957659634760054
```

```
tree.score(val_df, val_target)
```

```
0.9846524249995863
```

From DecisionTree I got Accuracy of 98% on validation data set

RandomForestClassifier

```
from sklearn.ensemble import RandomForestClassifier
```

```
model=RandomForestClassifier(n_jobs=-1, random_state=42)
model.fit(train_df, train_target)
```

```
RandomForestClassifier(n_jobs=-1, random_state=42)
```

```
model.score(train_df, train_target)
```

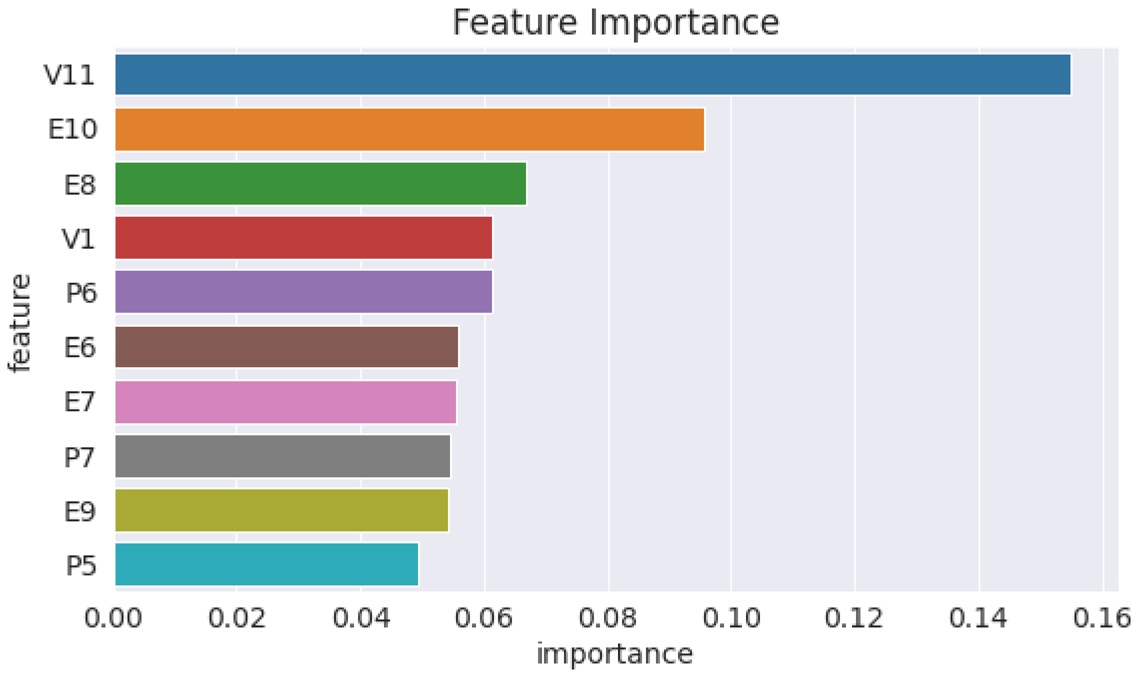
```
1.0
```

```
model.score(val_df, val_target)
```

```
0.9944483973987722
```

```
importance_df = pd.DataFrame({
    'feature': train_df.columns,
    'importance': model.feature_importances_
}).sort_values('importance', ascending=False)
```

```
plt.title('Feature Importance')
sns.barplot(data=importance_df.head(10), x='importance', y='feature');
```



```
val_probs = model.predict_proba(val_df)
val_probs
```

```
array([[1.  , 0.  ],
       [0.  , 1.  ],
       [0.01, 0.99],
       ...,
       [0.98, 0.02],
       [0.  , 1.  ],
       [0.  , 1.  ]])
```

```
def test_params(**params):
    model = RandomForestClassifier(random_state=42, n_jobs=-1, **params).fit(train_df,
    return model.score(train_df, train_target), model.score(val_df, val_target)
```

```
test_params(max_depth=26)
```

```
(0.9962106717577146, 0.9873827213608459)
```

```
test_params(max_leaf_nodes=2**20)
```

```
(1.0, 0.9946717852828753)
```

```
test_params(max_features='log2')
```

```
(1.0, 0.9940595370079262)
```

```
test_params(min_samples_split=100, min_samples_leaf=60)
```

```
(0.952846443264531, 0.9500024820876012)
```

```
test_params(bootstrap=False)
```

```
(1.0, 0.9967319179918257)
```

```
test_params(class_weight='balanced')
```

```
(1.0, 0.9946055962801781)
```

```
test_params(n_estimators=200)
```

```
(1.0, 0.9947048797842238)
```

```
jovian.commit()
```

```
[jovian] Detected Colab notebook...
[jovian] Uploading colab notebook to Jovian...
Committed successfully! https://jovian.ai/btech60309-19/driver-alertness-detection
```

```
'https://jovian.ai/btech60309-19/driver-alertness-detection'
```

From RandomForestClassifier I got Accuracy of 99% on validation data set

xgboost

```
! pip install xgboost --upgrade --quiet
from xgboost import XGBClassifier
```

```
model=XGBClassifier(random_state=42, n_jobs=-1, n_estimators=100,max_depth=10,learning_
model.fit(train_df, train_target)
```

```
/usr/local/lib/python3.7/dist-packages/xgboost/sklearn.py:1224: UserWarning:

The use of label encoder in XGBClassifier is deprecated and will be removed in a future release. To remove this warning, do the following: 1) Pass option use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].


[10:45:12] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default evaluation metric used with the objective 'binary:logistic' was changed from 'error' to 'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

```
XGBClassifier(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
              gamma=0, gpu_id=-1, importance_type=None,
              interaction_constraints='', learning_rate=0.3, max_delta_step=0,
              max_depth=10, min_child_weight=1, missing=nan,
              monotone_constraints='()', n_estimators=100, n_jobs=-1,
              num_parallel_tree=1, predictor='auto', random_state=42,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, subsample=1,
              tree_method='exact', validate_parameters=1, verbosity=None)
```

```
model.score(train_df, train_target)
```

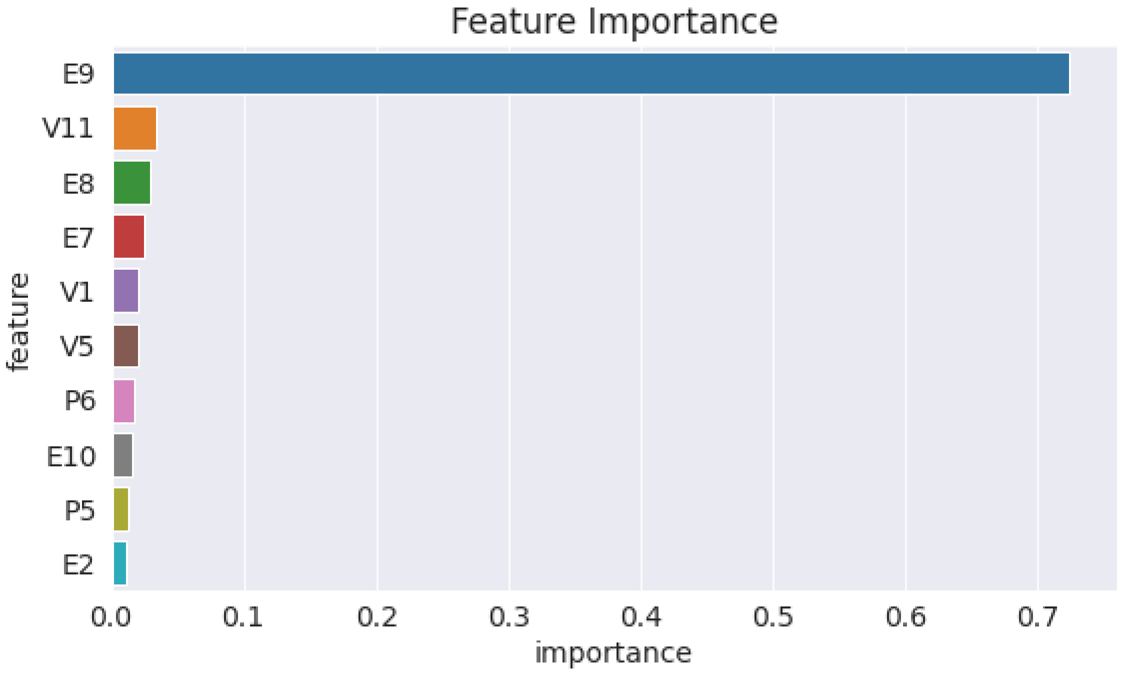
```
0.9923634280182765
```

```
model.score(val_df,val_target)
```

```
0.9842966591100889
```

```
importance_df = pd.DataFrame({
    'feature': train_df.columns,
    'importance': model.feature_importances_
}).sort_values('importance', ascending=False)
```

```
plt.title('Feature Importance')
sns.barplot(data=importance_df.head(10), x='importance', y='feature');
```



```
model=XGBClassifier(random_state=42, n_jobs=-1, n_estimators=100,max_depth=10,learning_
model.fit(train_df, train_target)
model.score(train_df, train_target),model.score(val_df,val_target)
```

```
/usr/local/lib/python3.7/dist-packages/xgboost/sklearn.py:1224: UserWarning:

The use of label encoder in XGBClassifier is deprecated and will be removed in a future
release. To remove this warning, do the following: 1) Pass option
use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your
labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].


[10:51:21] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default
evaluation metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

```
(0.9923634280182765, 0.9842966591100889)
```

```
model=XGBClassifier(random_state=42, n_jobs=-1, n_estimators=200,max_depth=20,learning_
model.fit(train_df, train_target)
model.score(train_df, train_target),model.score(val_df,val_target)
```

```
/usr/local/lib/python3.7/dist-packages/xgboost/sklearn.py:1224: UserWarning:

The use of label encoder in XGBClassifier is deprecated and will be removed in a future
release. To remove this warning, do the following: 1) Pass option
use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your
labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].


[10:55:25] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default
evaluation metric used with the objective 'binary:logistic' was changed from 'error' to
```

```
'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

```
(0.9969801205056023, 0.9892194661856932)
```

```
model=XGBClassifier(random_state=42, n_jobs=-1, n_estimators=300,max_depth=30,learning_
model.fit(train_df, train_target)
model.score(train_df, train_target),model.score(val_df,val_target)
```

```
/usr/local/lib/python3.7/dist-packages/xgboost/sklearn.py:1224: UserWarning:

The use of label encoder in XGBClassifier is deprecated and will be removed in a future
release. To remove this warning, do the following: 1) Pass option
use_label_encoder=False when constructing XGBClassifier object; and 2) Encode your
labels (y) as integers starting with 0, i.e. 0, 1, 2, ..., [num_class - 1].


[11:11:40] WARNING: ../src/learner.cc:1115: Starting in XGBoost 1.3.0, the default
evaluation metric used with the objective 'binary:logistic' was changed from 'error' to
'logloss'. Explicitly set eval_metric if you'd like to restore the old behavior.
```

```
(0.9931059874282003, 0.9826915757946817)
```

```
model.feature_importances_
```

```
array([3.4913863e-04, 3.0822146e-05, 1.7161058e-04, 0.0000000e+00,
       1.0994467e-03, 1.2808032e-03, 0.0000000e+00, 0.0000000e+00,
       3.8135349e-04, 1.0570502e-03, 2.4838606e-04, 2.0420022e-04,
       4.2206497e-04, 5.4727099e-04, 2.4867286e-03, 4.2317216e-03,
       9.7612596e-01, 1.3276930e-03, 3.1447748e-04, 1.4270708e-03,
       4.0409621e-04, 1.4568334e-04, 6.2006008e-04, 2.4397476e-03,
       5.7599269e-04, 0.0000000e+00, 2.0909356e-04, 0.0000000e+00,
       6.5506744e-04, 3.2446852e-03], dtype=float32)
```

From XGBOOST I got Accuracy of 98% on validation data set

Finally I Selected RandomForest

```
model=RandomForestClassifier(n_jobs=-1, random_state=42,bootstrap=False)
```

```
target=fordtrain['IsAlert']
model.fit(x_train,target)
```

```
RandomForestClassifier(bootstrap=False, n_jobs=-1, random_state=42)
```

```
model.score(x_train,target)
```

```
1.0
```

```
preds=model.predict(x_test)
```

```
submission
```

	TrialID	ObsNum	Prediction
0	0	0	0
1	0	1	0
2	0	2	0
3	0	3	0
4	0	4	0
...	...	...	...
120835	99	1206	0
120836	99	1207	0
120837	99	1208	0
120838	99	1209	0
120839	99	1210	0

120840 rows × 3 columns

```
submission['Prediction']=preds
```

```
submission
```

	TrialID	ObsNum	Prediction
0	0	0	1
1	0	1	1
2	0	2	1
3	0	3	1
4	0	4	1
...	...	...	...
120835	99	1206	1
120836	99	1207	1
120837	99	1208	1
120838	99	1209	1
120839	99	1210	1

120840 rows × 3 columns

```
jovian.commit()
```

```
[jovian] Detected Colab notebook...
[jovian] Uploading colab notebook to Jovian...
Committed successfully! https://jovian.ai/btech60309-19/driver-alertness-detection
```

```
'https://jovian.ai/btech60309-19/driver-alertness-detection'
```

```
accuracy_score(solution.Prediction,preds)
```

```
0.8239904005296259
```

```
submission.to_csv('final_result.csv', index=None)
```

At The End I Got Accuracy Of 99% on Validation Data Set and 83% on Test Data Set.

```
import os
os.listdir('./')
```

```
['.config', 'stayalert', 'final_result.csv', 'sample_data']
```

```
from IPython.display import FileLink
```

```
# Doesn't work on Colab, use the file browser instead.
FileLink('final_result.csv')
```

final_result.csv

Saving The Model

```
import joblib
```

```
stay_alert = {
    'model': model,
    'imputer': imputer,
    'scaler': scaler,
    'encoder': encoder,
    'input_cols': input,
    'target_col': output,
    'numeric_cols': input,
    'categorical_cols': cat_cols,
    'encoded_cols': encoded_cols
}
```

```
joblib.dump(stay_alert, 'stay_alert.joblib')
```

```
jovian.commit()
```

```
[jovian] Updating notebook "btech60309-19/driver-alertness-detection" on
https://jovian.ai
```

```
[jovian] Committed successfully! https://jovian.ai/btech60309-19/driver-alertness-detection
```

```
'https://jovian.ai/btech60309-19/driver-alertness-detection'
```

Summary and References

Finally I got Accuracy of 83% on Test set.

YOU can download the Dataset from Kaggle:

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

```
jovian.commit()
```